'68' MICRO JOURNAL

$2.95USA

Australia	A $ 4.75	New Zealand	NZ $ 6.50
Singapore	S $ 9.45	Hong Kong	H $23.50
Malaysia	M $ 9.45	Sweden	30.-SEK

VOLUME V ISSUE X • Devoted to the 68XX User • October 1983

"Small Computers Doing Big Things"

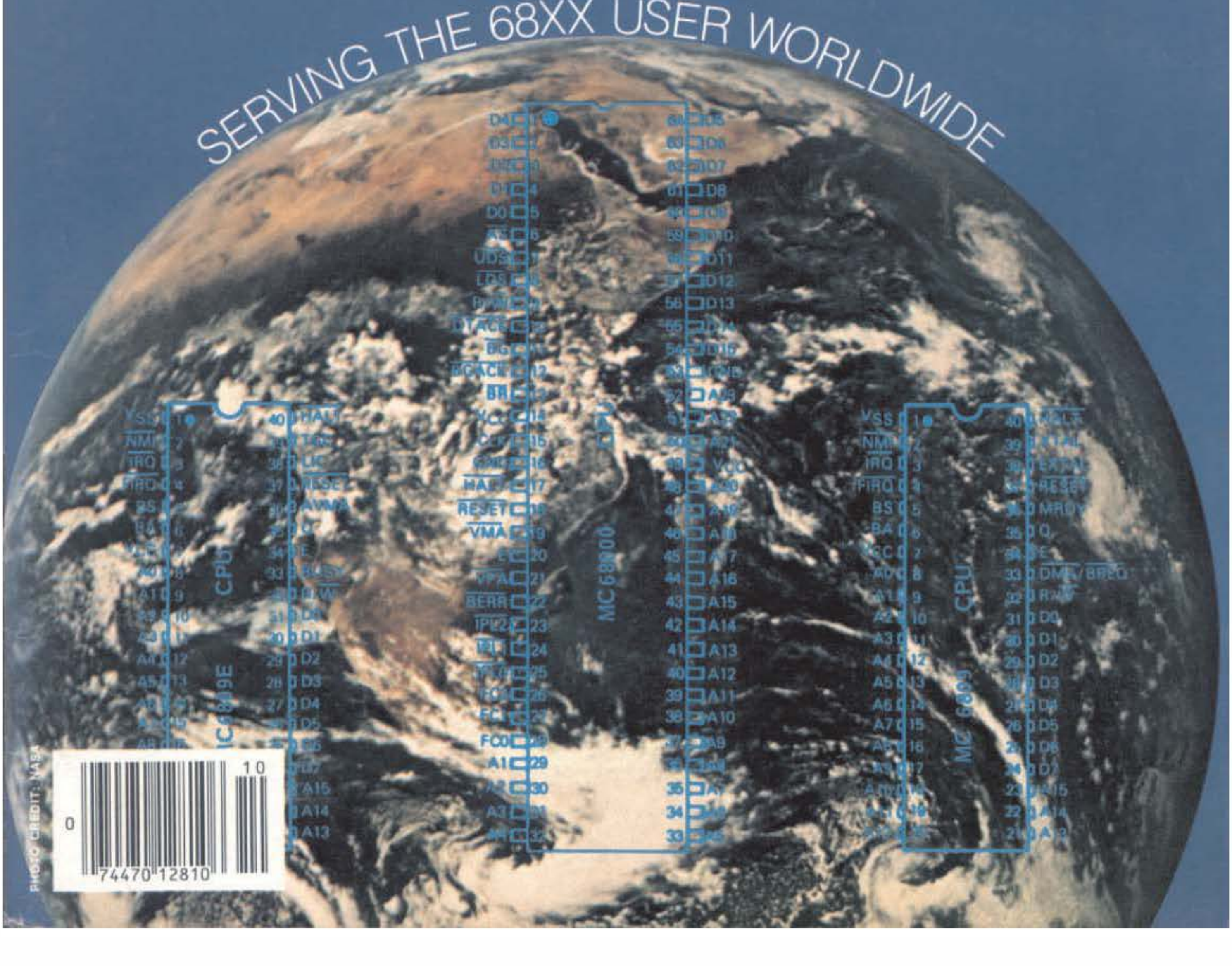
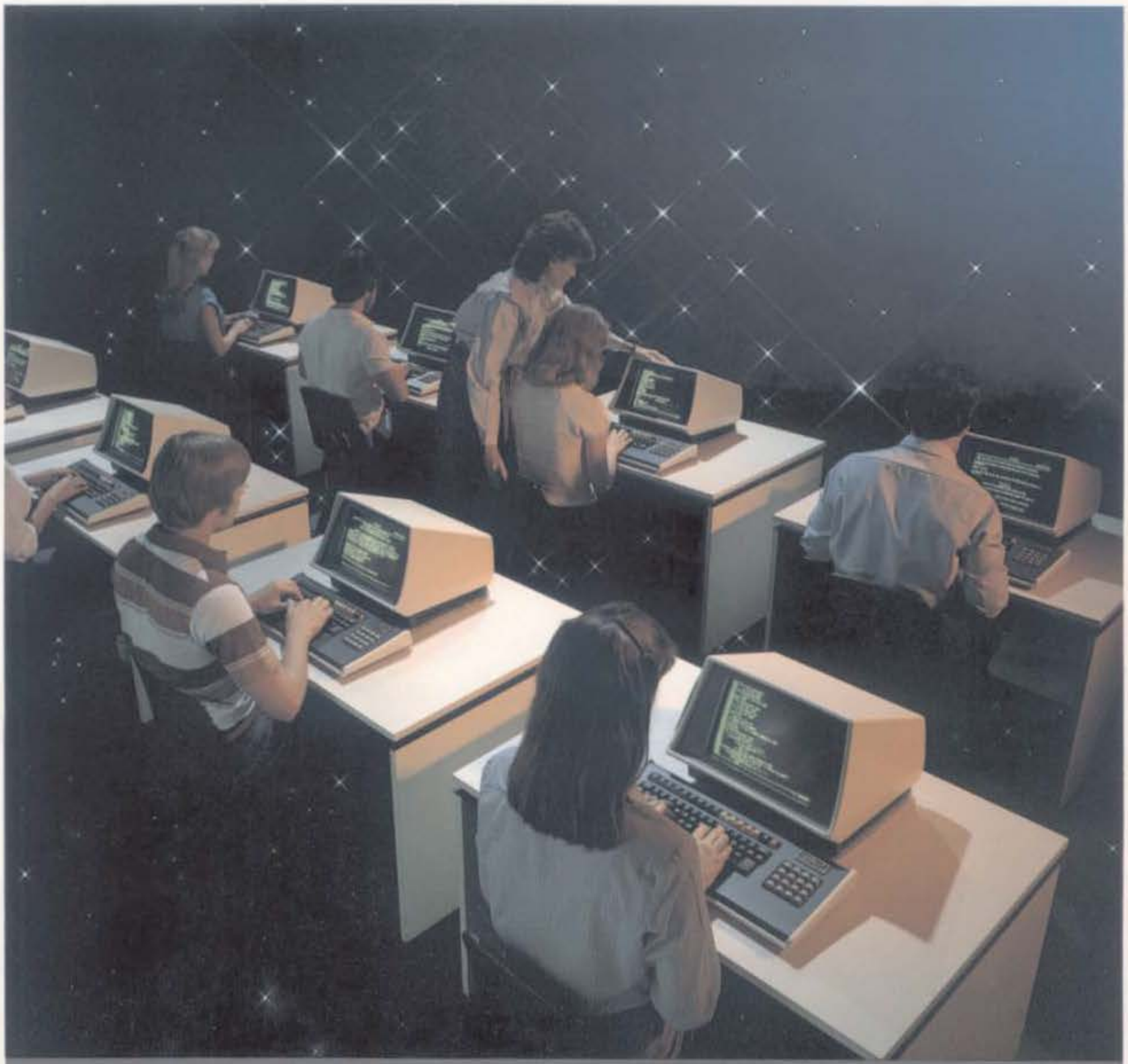


PHOTO CREDIT: NASA



FREE FROM LIMITATIONS, A TEACHER CAN UNLOCK IMAGINATIONS.

By 1985, fully 75% of all jobs will involve computers. So for today's teachers, selecting the right computer is an awesome responsibility. It's a decision you can feel confident about with Southwest Technical Products Corporation's S + computer system. The S + is specially designed for a student's tomorrow and a teacher's today. It's built tough. It's simple to operate. You can teach without becoming a hardware specialist.

The S + is fluent in a variety of languages: BASIC, PASCAL, COBOL, FORTRAN, PILOT, FORTH, "C," and MUMPS and allows you simultaneous use in the classroom. This flexibility enables you to use your own curriculum, not a machine's version. The system also allows for individual learning styles. Students go as fast and as far as the teacher chooses.

Even though it's priced like a smaller personal computer, the S + has the multi-task power and expansion abilities of a professional computer. It can grow with you and your students.

Tomorrow is in your hands, today. Call Fisher Scientific, exclusive distributor of our S + system, and you'll get the key to unlock imaginations.

For more information.
Call toll-free 800-621-4769.
Illinois customers call collect
(312) 378-7770.

Fisher Scientific
An ALLIED Company

Educational Materials Division
4901 W. LeMoyne St.,
Chicago, Illinois 60651

THE COMPLETE 6809

C

The Microware C
is a fully implemented
Kernighan and Ritchie
standard C Compiler
with relocatable
assembler, linker, profiler,
UNIX* and OS-9*
standard libraries.

Suggested U.S.
retail price $400.

Now available from

MICROWARE®
5835 Grand Avenue
Des Moines, Iowa 50312
515-279-8844
Telex: 910-520-2535

*UNIX is a trademark of Bell Laboratories. OS-9 is a trademark of Microware and Motorola, Inc.

'68' MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished herd/software.

COMPUTERS-HARDWARE

Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

-

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

-

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

Editorial Staff

Don Williams Sr.	Publisher
Larry E. Williams	Executive Editor
Tom E. Williams	Production Editor
Robert (Bob) Nay	Color Editor

Administrative Staff

Mary Robertson	Office Manager
Joyce Williams	Accounting
Carolyn Williams	Subscriptions
Penny Williams	File Management

Contributing Editors

Ron Anderson
Norm Commo
Peter Dibble
Dr. Theo Elbert
William E. Fisher
Dr. E.M. Pass

Special Technical Projects

Clay Abrams K6AEP
Tom Hunt

CONTENTS

Vol. V, Issue X OCTOBER '83

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson. Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 52

Items Submitted for Publication

Articles submitted for publication should be **accompanied by the authors full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the ,pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submitted articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

The GIMIX CLASSY CHASSIS 6800/6809 SS-50 Bus Mainframe

The CLASSY CHASSIS includes:

A heavyweight, aluminium cabinet (18" wide x 21" deep x 7" high) painted in a putty colored, durable baked enamel finish. The cabinet holds our 6800/6809 mother board, CV Ferro-resonant power supply, and has provisions for mounting one or two 5¼" Floppy or Winchester disk drives. The back panel is punched for 15 "D" type data connectors (25 pin) and has provisions for two removable connector plates that are available in a variety of connector configurations. Cabinets are normally supplied with two blank plates unless other types are required or specified. The cabinet includes a fan and ventilation slots which direct cooling air over the boards and power supply. The front panel has a 3 position, key locking, power switch that permits the reset switch to be locked out, preventing accidental system reset, and a three position Reset/Abort switch. Optional filler plates are available for systems that do not use the 5¼" drive openings.

The 6800/6809 SS-50/C Motherboard

This highly versatile motherboard is easily reconfigured for a variety of 6800 and 6809, SS-50 and SS-50C bus configurations.

Gold Plated connectors are used throughout to insure long lasting electrical contact and protection against corrosion.

It has fifteen 50 pin slots, 8 DIP-switch addressable 30 pin I/O slots, and a special 10 pin slot for the baud rate generator board. The fully buffered I/O block can be configured for 4, 8, or 16 decoded addresses per slot, and is DIP-switch addressable to any 32, 64, or 128 byte boundary. Extended address decoding (SS-50C) allows the I/O block to be addressed anywhere in the 1M byte address space.

The baud rate generator board provides 11 standard (16X) baud rates, from 75 to 38.4K in 2 groups. Programming jumpers allow easy selection of up to five baud rates. The five baud rate lines on the 50 pin bus are easily disconnected from the 30 pin bus for use with SS-50C extended addressing or as user defined lines. A slow I/O circuit for the 6809 CPU can be used to generate an MRDY signal whenever an I/O slot is accessed (this allows, for example, using PIO Disk Controllers with a 2MHz 6809 CPU).

All data, address, and control lines are fully terminated and separated by noise reducing ground lines on the bottom of the board.

The .090" thick, double sided P.C. board has a full ground plane Faraday Shield on the top side to further reduce noise.

Industrial Quality Power Supply

Features:

A Constant Voltage, Ferro-resonant, Faraday shielded transformer, custom designed for GIMIX, for brown-out and overvoltage protection and for proper system operation, even under adverse AC power input conditions. It includes an AC line filter, 3 DC filter capacitors, and a filter assembly board with clamping terminal block for easy wiring. The power supply provides +8 Volts at 30 Amps, +16 Volts at 5 Amps, and −16 Volts at 5 Amps, enough to power a fully loaded system plus the 5¼" disk drives (floppy and/or Winchester) that can be installed in the cabinet. All outputs are filtered and individually fused. The standard version operates over an AC input range of 90 to 140 Volts, 60 Hz. Export versions are available for inputs of 95 to 130 or 190 to 260 volts, 50 Hz.

GIMIX 6809 systems support five predominant operating systems:

OS-9 GMX III,
OS-9 GMX II,
UniFLEX,
OS-9 GMX I,
FLEX

and a wide variety of languages and development software.

Whatever your application: software development, instrumentation, process control, educational, scientific or business; whether you need single or multi-user capabilities, GIMIX has hardware and the operating systems to get the job done reliably.

Please phone or write if you need further information.

GIMIX inc.

1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609 • (312) 927-5510 • TWX 910-221-4055

Build performance into your system

with OS-9™ software tools

Unix*-based, multitasking, modular, and versatile: these key features are some of the reasons why more 6809 computer manufacturers have selected OS-9 as their standard operating system than any other. And OS-9 has been put to work by thousands of users in almost every conceivable computer application in business, science, industry, education, and government.

Your operating system should not be a barrier between you and your computer. OS-9 is very friendly and easy to use. Its modular structure makes it easy to customize, plus its comprehensive documentation shows you exactly how to interface it to just about any I/O device.

OS-9's advanced features unleash the performance potential of almost any 6809 computer — large or small. In many respects the OS-9/6809 combination is more powerful than many *minicomputers!*

There are two basic versions of OS-9. Both have the same basic features and capabilities. OS-9 Level One runs on small to medium sized systems having up to 64K memory. The Level Two version runs on medium to large size systems having memory management hardware and up to 1 megabyte of memory, and includes record and file locking for multiuser database applications.

Here are just a few reasons why you should insist on OS-9 for your microcomputer system.

- Over 40 utility commands
- Friendly "Shell" command interpreter
- Tree-structured multilevel file directories
- Full timesharing support with log-in and file security
- Fast, secure random and sequential access files
- Comprehensive English language error messages
- Compact real-time multitasking executive
- Hardware or software memory management
- Device independent interrupt-driven I/O
- Fully ROMable for small control systems
- Standard versions available from manufacturers of most popular 6809 computers

OS-9 PASCAL Language Compiler

- most complete and versatile PASCAL available for the 6809
- capable of generating P-code for interpretive execution while debugging OR
- highly optimized 6809 assembly language source code output for maximum speed
- "virtual memory" P-code interpreter lets you run large PASCAL programs

CIS COBOL ***Compiler

- ideal for most demanding business applications
- features ISAM, Debug, ACCEPT/DISPLAY, and Interprogram Communications modules
- retains full compatibility with CP/M software
- meets ANSI 1974 Level One COBOL standard and is GSA certified
- Also available-FORMS 2 automatic program generator for easy interactive design of screen oriented applications

BASIC09 Structured Basic Interactive Compiler**

- fastest and most comprehensive full Basic language available for the 6809
- combines standard Basic with the best features of PASCAL
- features compiler speed, interpreter friendliness and superlative debugging facilities
- option available: Run B...a ROMable run-time system for compiled Basic 09

C Language Compiler

- complete implementation of the UNIX version 7 C language
- Includes INT, CHAR, SIGNED, UNSIGNED, FLOAT AND LONG data types, structures, unions, standard C library and a full preprocessor with macro definitions
- generates fully reentrant 6809 assembly language source code output

For more information contact your computer supplier, or

MICROWARE

Microware Systems Corporation
5835 Grand Avenue, Des Moines,
Iowa 50312 515-279-8844 • Telex
910-520-2535

*Unix is a trademark of Bell Laboratories. ***CIS Cobol is a trademark of Micro Focus, Inc. OS-9** and Basic09 are trademarks of Microware and Motorola, Inc.

FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

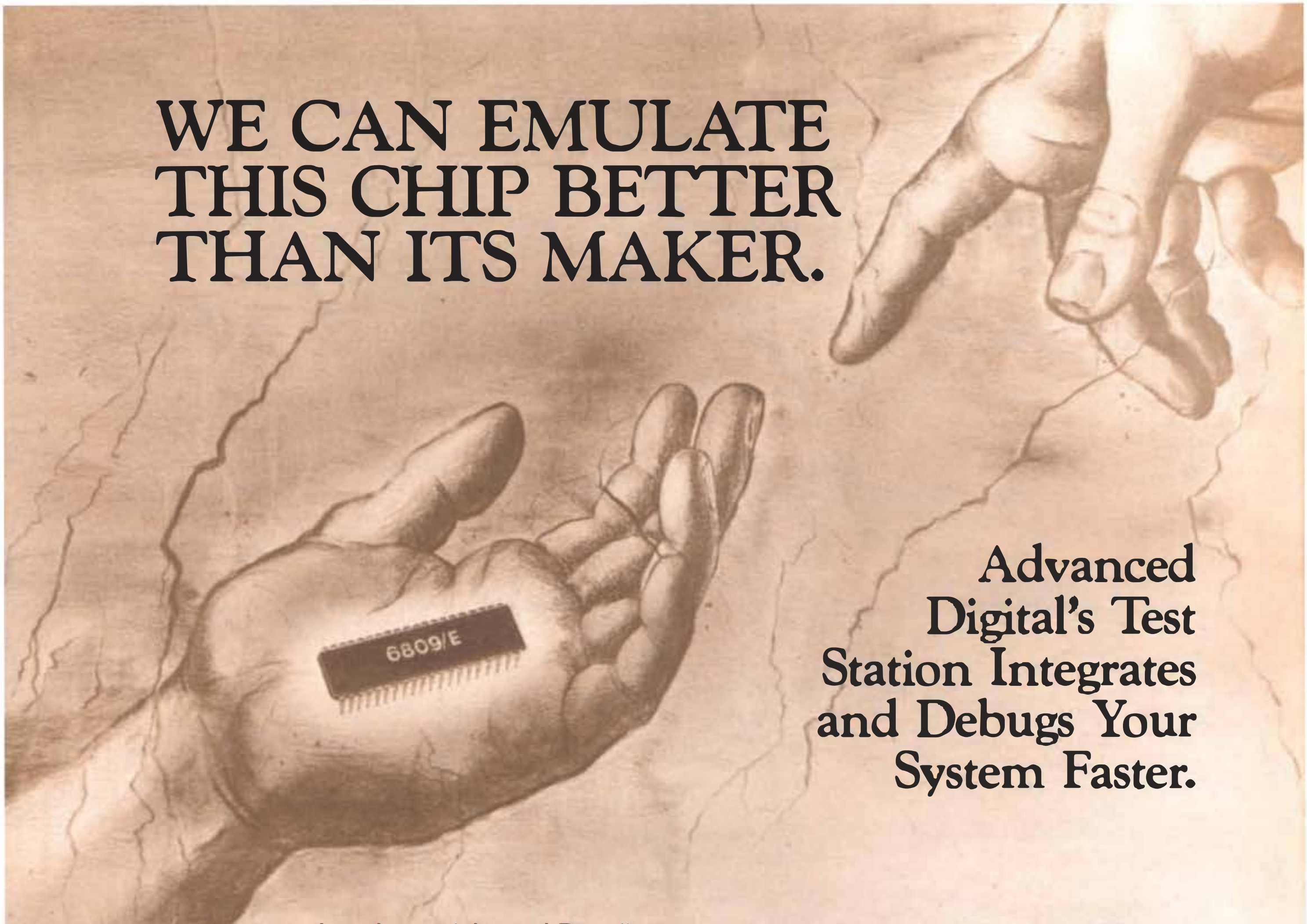
With disk: 8" **$22.90** + $2.50 S/H

See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

™FLEX is a trademark of Technical Systems Consultants

WE CAN EMULATE THIS CHIP BETTER THAN ITS MAKER.



Advanced Digital's Test Station Integrates and Debugs Your System Faster.

Introducing Advanced Digital's new 4009B Test Station. The test station that combines emulation and logic analysis into one system. Until now, most emulators operated in real-time by restraining the target system. Not with Advanced Digital. Our 4009B Test Station is totally transparent. Totally real-time. We meet or exceed manufacturer's timing specifications. This is particularly important when emulating interrupt intensive systems and accommodating DMA operation.

Universal operation enables you to emulate 6809 and 6809E series microprocessors without making software or hardware changes. Up to 128K of memory overlay, four hardware triggers and 2K deep of trace help you pinpoint problems faster. A line-by-line assembler and a powerful trace disassembler are standard. They enable you to view software and make changes without returning to the development system.

4009B TEST STATION

MORE THAN AN EMULATOR

As a "digital engineering test station," Advanced Digital's 4009B gives you a synergistic approach to integration problems. Two microprocessors control the system. One is dedicated to emulation while the second microprocessor supervises the 32 channel logic analyzer and all system menus. This design lets you emulate and perform all system functions simultaneously.

Discover the most powerful diagnostic tool available for hardware and software integration.

Call Advanced Digital Technology for a free demonstration.

Advanced
Digital
Technology

13400 Northup Way, #27
Bellevue, WA 98005
(206) 643-2382

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted. "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.

ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it super-fast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.

ONLY $500

TMP FreeForm fills the void between a structured DBMS and word-processing; it's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.

ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.

NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

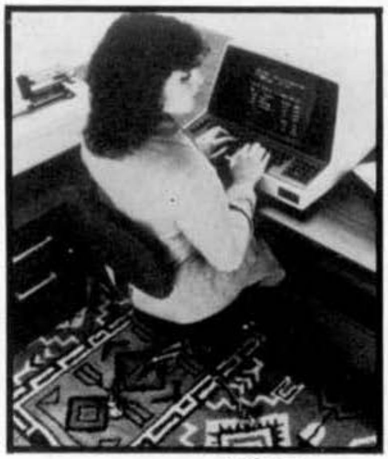
AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*



Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up; perfomance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

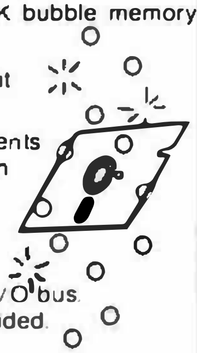
SMOKE SIGNAL®
Chieftain Computers
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

BUBBLE MEMORY

HIGH RELIABILITY STORAGE FOR THE SS-50 BUS

DISKBUB provides 128K bubble memory replacing a disk drive.

- —no moving parts to wear out
- —direct boot capability
- —withstands harsh environments such as dust, heat, vibration
- —faster than standard drives
- —utility software supplied



Diskbub plugs into the 30 pin I/O bus. Flex09 drivers and boot rom provided. Available for only

$845.00

UNIVERSAL DATA RESEARCH INC.

2457 Wehrle Drive, B-68
Buffalo, New York 14221
Phone (716) 631-3011

Dealer Inquiries Welcome

Business Software For The

COLOR COMPUTER

WITH FLEX* & XBASIC

Data Base Manager

Part I	$ 49.95
Part II*	$ 49.95
Church Contributions	$ 49.95
Balanced Billing System	$ 49.95
Single Entry General Ledger	$ 49.95

Integrated Business Software*

*available for Color Computer only

Accounts Receivable	$ 99.95
Accounts Payable	$ 99.95
General Ledger	$189.00
Inventory 2	$ 69.00
Payroll	$ 99.95
64K memory upgrade including installation	$ 79.00

UNIVERSAL DATA RESEARCH INC.

Call or Write for free catalogue

2457 Wehrle Dr., D-80, Buffalo, NY 14221
PHONE (716) 631-3011

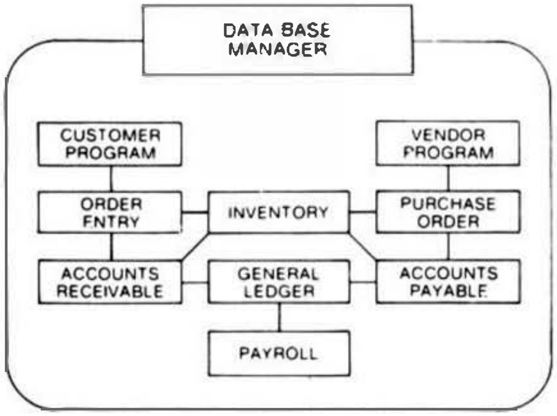
Dealer Inquires Welcome

-417

New, Lower Prices

FLEX* and UniFLEX*

software for the 6809



Integrated Business Programs

	FLEX	UniFLEX
Accounts Payable	$295	$395
Accounts Receivable	$295	$395
General Ledger	$295	$395
Inventory 2	$295	$395
Payroll	$295	$395
Data Base Manager	$350	$450
Word Processing		
Software	$295	$395
WP Menu		$150
P Control		$150

UNIVERSAL DATA RESEARCH INC.

*FLEX & UniFLEX are Trademarks of Technical Systems

2457 Wehrle Drive, D-68, Buffalo, NY 14221
PHONE (716) 631-3011

Dealer Inquires Welcome

VISA

Color Micro Journal™

5900 Cassandra Smith Rd. ★ Hixson,TN. 37343

COLOR MICRO JOURNAL™ Is A Monthly Tabloid Publication for Color Computer USERS!

★★

COLOR MICRO JOURNAL™ is a Magazine FOR Color Computer Users BY Color Computer Users. Columns on various compatible Operating Systems, Languages, Uses (Bulletin Boards, Clubs, using the RS BASIC, and so on), etc.

★★★

Programs - Games - Reviews - Education - Hardware - Software - New Product Announcements - Books

★★★★

Get the MOST from your COLOR COMPUTER
without being an Engineer.
DON'T MISS A SINGLE ISSUE
Subscription Rate of only $16.50 a Year!!!

★★★★★

COLOR MICRO JOURNAL™ published by the ONLY pure 68xx, INTERNATIONAL Computer Magazine. '68 Micro Journal has provided coverage for over FIVE Years. We KNOW the Color Computer, the Software (both FUN and WORK) that IS and CAN BE run on it. We KNOW the products that ARE, CAN BE, or WILL BE used on the Color Computer.

For Ordering Subscriptions By Phone
Call 1-800-338-6800

Color Micro Journal

Limited Time Charter Rates

USA - $16.50 per year, Canada & Mexico - $23.00 per year

Surface Foreign - $28.00 per year, Airmail Foreign - $52.00 per year

™Color Micro Journal is a trademark of Computer Publishing Inc.

Yes! Start my copy of Color Micro Journal coming as soon as possible!

Name________________________________

Address______________________________

City____________ **State**______ **Zip**________

□ **Visa** □ **Master Card** □ **Check or Money Order Enclosed**

Card #______________________________

Exp/Date______

Rumors & Such

This time there is little if any 'rumor' in this part of my occasional space grabbing. However, there is a situation occurring in other magazines, and especially club newsletters and bulletins, that I have some pretty strong feelings about. That is software pirates, or in other words - **thieves!**

Over the years we have always had the problem of shared software purchases. By this I mean some individual buys the software and then shares it with a friend. Maybe the friend helps with the purchase price or maybe not, however, it has deprived the vendor of a potential sales. "Not much", you say, well maybe but if it happens often enough the vendor or programmer can't make ends meet, goes out of business and we have lost another source of software. I wonder just how useful your computer would be without **any** software, excepting that you can or will (have already) write? Not very I suspect!

Occasionally I get a phone call or letter from some programmer or software vendor complaining about losses due to thief. Many just accept it as a normal way of life and continue on. Others moan and groan but really do nothing. While others either 'intend' to do something (someday) or go off half cocked and blow the deal. Time passes, software continues to be stolen, right, **stolen**, and nothing has really been accomplished to get it **STOPPED**. In a few, very few, cases that I am aware of has anything been done. One I recanted to you a few months back. That instance was handled in a proper and legal way and it was nipped. However, it was never determined that all the damage had been corrected. Just that local practice curtailed. Local in nature and impact. Worldwide it continues to grow and flourish.

Some software vendors have decided to ignore the situation and raise the cost of their software to compensate for all potential losses. So, who pays? Sure you know, you and I. I don't like it! You shouldn't either unless your wallet is a lot thicker than mine (which isn't very). Others ignore it also and go bankrupt! While still others ponder what to do. Very few have taken the time or been willing to expand the effort to take some sort of effective action, even when they have had the chance. For them they are getting what they deserve! For those few who have expanded time, effort and funds to attempt to do something, in many cases it was useless. Not all, but mostly. All because of indifference, and a general lack of understanding of the problem, on the part of all sides, the offender, the offended and sadly the courts. So again we all loose!

"Well, ho hum, what brought this up again?", you may be asking.

"A couple of sheets of paper", I reply.

This past weeks mail brought a brown manila envelope with two (2), yes two (2), like in T-W-O, advertising contracts, prepaid - yep, prepaid for one entire year for both **68 MICRO JOURNAL and COLOR MICRO JOURNAL**. Now you cannot imagine how utterly delighted and overjoyed we become if we get just one (1) contract for a full page, for a 12 month period for even one of our magazines. Prepaid yet, "too good to be true!!!". This was double delight - until I looked at the advertising copy.

The copy is professionally done, two color (more expensive) and starts out with a large picture of a helping hand. The first few lines of text says "Now help is here for the beleaguered computer users", good so far. But the next line is the kicker, " A new and long needed computer users service is now available - **RENT and try before you BUY!"**.

A few 'library' programs were indicated, none of any worth, BUT it goes on to explain that if there is any other software packages needing to be rented they would secure the software and have it available, in short order. The plan was for the user to subscribe for a modest fee ($50) to the rental library plan and then rent ANY software for approximately 20-30% of it's retail selling price. It goes on to explain that you are not to 'copy or reproduce' the rented software (you must even sign a statement to that effect when signing up) and that you have only 30 days to make a decision. So you could 'rent' the software to 'test' and then if interested they would sell it to you, at a discount. I find nothing wrong with the discount part, but the rest....

With a sigh I sent it back, along with the two cashiers checks, explaining that we felt it was at the best a poor if not terrible plan. And that under no circumstances would we sell them advertising space for that particular ad, or plan. Twentyfour (24) full page ads down the tube, and especially at a time when we could use MORE advertising. Prepaid yet!

I wonder what would (or will) happen when these ads start to appear in some of the larger magazines. What will you software vendors do? "Now getting out of hand," you reply. "Right," says me, "but what about last year, and the year before?". If you all might remember I offered to help fund, and it will take funds, a kitty to start a concerted effort against this and other thief activities. Boy, I got all of about three calls and five or six letters, most willing to do something with a promise to 'get back to you'. The 'getting back' part is still hanging. Maybe I should have accepted the two ads.

For a couple who really seemed interested, one has quit business. He had a pretty good product and I guess there were a lot of other factors involved, but I am sure he felt that it just wasn't worth the ratio of effort and return.

What occurred to cause me to write this is that a few days ago I received a phone call from the ad agency for the outfit 'renting' and selling software. I told them in no uncertain language what my true feelings were. Of course they really didn't understand because their bag is developing and selling advertising schemes. I was even informed that I legally could not refuse their ads. Well maybe, as I have been through this before with another large outfit and they ended up paying our lawyers for the time and effort spent getting us off the hook. There is still such a thing as - right or wrong! Seems that no one, except those in the computer business, knows the difference when it comes to stealing software, by one means or another. And some of them don't.

This morning I received another call concerning those ads. This time from the 'president' of the company. Now he knows the difference. Told me so. But his general excuse was, "well it is being done all the time" and something to the effect that it would go on, despite what I might think or do. I agree! However, I don't have to participate. Seems that he feels some of the other magazines 'will' accept his ads. Gathered that he was meeting some resistance, but not entirely. His parting remarks was that after he got in some other magazines (told me how much bigger they were) then they would contact me again. He probably will - but not **here!**

As I read back over this piece I cannot see why I write this. Guess I just have to get it off my chest. If this sort of thing takes hold and is legally upheld, then good-by good software. Nobody is going to write it for nothing. Then what will we all have? Huh?

DMW

Flex User Notes

Ronald W. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

A REAL ARITHMETIC PROBLEM

The June '68' Micro Journal contains a letter from Jim Schreier of Phoenix Arizona, describing a "bug" in TSC BASIC. This discussion comes up every now and then in the computer magazines, and as always, old Ron is here with his two cents worth. First I have to say that nearly every good programming book contains a caution about using equality or inequality tests with REAL variables. I'll attempt to explain why, and show the proper way to program such tests (so they will work in anybody's language).

The sticky part of the problem is that such tests may or may not work in different implementations of the same language. This is because each of the implementations code their floating point math routines slightly differently. Implementations that use BCD math (binary coded decimal) may sometimes (though not always) produce results that are more accurate than the implementations that use binary arithmetic. The BCD versions are always slower in calculations than the binary versions.

The simplest analogy that I can think of, to describe the problem is the infinitely repeating decimal fraction. Suppose we decide that we are to have a 6 digit arithmetic. Now divide 1 by 3 and the answer is .333333 --- with the rest of the infinite string of 3's thrown away. Now multiply that decimal fraction by 3 again and the result has to be .999999. If we use more digits to represent the number, we get back more 9's, but we can never get the original 1.00000000! If that is true, you are about to say, why do any of the languages produce the proper results for equality and inequality tests. That is a fair question. The answer is that some of the implementations carry their calculation to an extra digit but round before returning the result. Even that is not infallible. Try a loop (in any language that allows it) like this: FOR N=1 TO 2 STEP .01: PRINT N : NEXT N and see if the error doesn't "creep". The greater precision the arithmetic package has, the faster you will see error accumulate in the last digits. This little problem is the reason why Pascal doesn't let you use REAL variables for array indices. If the step doesn't happen to represent itself evenly in binary, you might accumulate quite an error.

In Pascal, the language prevents you from doing such a thing. You might code the same loop in it this way: FOR N := 800 TO 900 DO WRITELN (N/100:2:9);... Granted dividing by 100 is slower than adding .01, but you always start with an exact number. N would be integer, of course converted to real by the real divide (integer divide in pascal would be N OIV 100).

The caution I mentioned earlier, that is included in many of the better books on programming, is that you do not make comparisons of REAL values for program branching decisions on the basis of EQUAL or UNEQUAL. Mr. Schreier presented a sample program (Routine A) which I have modified slightly to make the output a bit easier to read. I added the output of the variable X as well. You may be surprised at what you get when you run this version of the program, (ERROR.BAS). The proper way to do an EQUAL or NOT EQUAL comparison for REAL numbers is to take the absolute value of the difference and test it for being less than (or greater than depending on the sense of the required result) a satisfactory error limit.

We seldom need to know that two numbers are equal to 17 digits precision. ERROR1.BAS is the same program with a comparison error limit of 1.0E-12, (ie. agreement to within 1 in the 12th digit). It works just fine. Since many other interpreters and compilers that allow REAL numbers, don't have 12 digit precision, you would most likely make the error limit something more in line with 1.0E-6, for a general program that would run in several implementations of several languages. I suspect that an allowed error of 1.0E-16 would work fine for TSC Extended BASIC.

Most compilers and interpreters won't balk when you ask them to compare REAL numbers, but they might well give you the wrong answer if you are not careful. The safest route is to assume that none of them will give you the correct answer for all cases, and use the technique described above.

Jim Schreier and Louis Boyd, I hope you won't feel unduely picked on by this discussion. You aren't the first programmers to have fallen into the trap of real number comparisons, and y   won't be the last. Just be advised that the floating point arithmetic packages in many of the compilers that are available would not pass your test.

MORE ON BUGS IN GENERAL

There are generally some cases in a language implementation, particularly when the software is new, that simply don't work correctly (or are not sufficiently explained in the documentation so that you can tell positively whether or not they work correctly.) When I run into such a case, as I recently did with a compiler not working properly for a multiple condition IF THEN with several statements ANDed together for the condition, I generally try to "program around" the difficulty. I try a different syntax, or write the statement in another way. For the case in point, I paired the AND conditions and set some intermediate variables TRUE or FA SE based on the ANDed pairs. Then I ANDed the intermediate results. etc. I also wrote a letter to the supplier of the compiler about the problem, and received a letter indicating that I had found a legitimate BUG and that it is being worked out. (I just received a repaired version).

In the line of writing this column, I have a chance to review a number of compilers that fall into the category of "new products". Frequently I find some combination that simply doesn't work as indicated in the manual (or as it ought to on a common sense judgement). Several times, I have received letters indicating that I had indeed found a real bug, and expressing surprise that no one else had reported the bug ("The product has been sold for a year now, and no one else has reported any problems"). More frequently than not, the supplier expresses thanks for the feedback. One supplier said something like this. "Either no one else is using our compiler to the extent that you have used it, or all our other customers are too lazy to report valid and serious bugs in a product for which they paid us good money."

The message should be clear. If you find a peculiar result in a program, try to extract a few lines of your program. Try a quick little test program to illustrate the bug. Remove all the extra code and send the supplier the minimum program that you can devise that will illustrate the malfunction. Most software suppliers will be extremely happy to receive the information you can provide. As compilers get more complex and capable, it becomes harder and harder to test them completely. One Hundred Percent testing is simply impossible. A few quick calculations will convince you that testing of a not terribly complex calculation for all possible values of the input variables, would take a VERY long time (generally something in the order of the astronomer's more or less agreed upon age of the universe). This is particularly true if the variables are REAL, and there are three or four involved in the calculation.

Another factor in software (compilers are not the only software vulnerable to such problems), being shipped with bugs, is the "new improved version". Many times, software authors get good ideas for improving or extending the usefulness of their compilers. Good examples might be adding REAL variables, implementing heap variables in Pascal, etc. Frequently, a product has been around and is working just fine. The author adds a new feature and forgets to do a complete and thorough test to see if the new feature clobbered any previously working part of the software. Sometimes, the new feature is too hastily tested. Speaking from long and sometimes sad personal experience, I can say that ANY change in a software package, no matter how innocent it looks, can introduce bugs of the first magnitude (and frequently does). Careful modular coding of software reduces the interaction between various "procedures" or "routines" and therefore reduces the chances of a change here clobbering something over there, but it still happens.

A few problems in a new software package, are common and to be expected. It is only when the number of bugs found are overwhelming, or when the supplier is unresponsive, or both, that we as software "consumers" have a right to complain. Get busy! If you have found a bug in your favorite software, write the supplier and tell him about it. Do yourself a favor first, though. Read the instruction manual carefully, and you may find that you had misinterpreted what should happen, and that the software works if you use it correctly.

When you write a software supplier, treat him with some respect. Don't say "Your #$("&$ compiler is a pile of "&%$$" (or words to that effect). Tell him "I wrote the enclosed three line program to allow me to enter a long integer variable and print it back out to the terminal. When I enter any positive number I get back what I entered. However, when I attempt to enter a negative number, all I get back is 0. I note that your compiler has the data type long unsigned integer, so I assume that long integers should be capable of having negative values. Am I doing something wrong, is my assumption incorrect, or have I found a legitimate bug in your compiler?"

When you write about a bug, d 't confuse the issue with a long list of "why did you" questions. "Why did you make me type .AND and .OR for those logical functions when compiler X lets me use & and ! for the same functions." In other words, don't mix bug reports with complaints about syntax requirements or "how it works" because the answer to those later questions will always be "That's how it works." Also report only bugs that you are about 90% certain about. If you report 12 bugs and the first 6 turn out to be misconceptions on your part, the supplier may never get to number 12 which happens to be a real and serious bug.

TSC has a little form that they send out with their software packages, called a "PROBLEM REPORT". After writing the above I happened to run across one. It pretty much seys what I did above with emphasis on "Make sure that you really have a problem." It then mentions something I didn't but should have. "Check out your hardware." I have had mysterious problems now and then with something that I know works perfectly, such as the TSC Assembler or editor. Such peculiar problems have always been found to be problems with memory. (I have an OLO SWTPc mother board and the tin plated pins don't always make contact, causing intermittent memory problems).

TSC recommends sending documentation of a bug in letter form so they can give it to their local "expert" on the software in question. Don't phone and demand an instant solution. Suppliers will first see if they can duplicate your problem, then find out what it is, and lastly spend some time figuring out the best solution. Many times, a patch of a few bytes will cure what looks like a very serious bug.

TIMING TESTS, LAST WORD

Two things happened after I mailed the last column to '68'. First I found a copy of January BYTE and verified my test programs for the benchmark. A careful reading of the article shows that the authors have Z-80 systems. They went into an evaluation of the available Pascal and "C" compilers for CP/M. They discussed how they optimized each program to run fastest with each particular compiler. In one case, that meant making all the variables Global, etc. The times presented for the 6809 compilers, obviously were those sent by readers who didn't spend a lot of time trying various methods to make the program run faster.

I reported a time of 13.5 seconds for PL9, but the next day I was reading the PL9 manual, and it mentioned that the order in which the variables are declared is important. They are all accessed indexed from one of the register pointers. The first are accessed with smaller offsets, which takes less clock cycles. I decided to try putting the index variables I,J,K,and L before the array declaration, and the program then ran in 12 seconds, for having moved one line of the source.

I was talking to Bud Pass on the phone last night, and the subject of the benchmarks came up. Bud was the second person to whom I've talked about these who obser d that the execution time of a program frequently depends more on the individual programmer than the language OR the algorithm.

```
5 REM ROUTINE A
6 REM INCORRECT DIVISION DEMO
10 DIGITS 17
20 FOR X=1 TO 20
30 N=1*X/X
40 M=1/X*X
50 PRINT X;TAB(20);N,M;
60 IF N<>M THEN PRINT " NOT EQUAL" ELSE PRINT
80 NEXT X

5 REM ROUTINE A
6 REM INCORRECT DIVISION DEMO
10 DIGITS 17
20 FOR X=1 TO 20
30 N=1*X/X
40 M=1/X*X
50 PRINT X;TAB(20);N,M;
60 IF ABS(N-M)>1E-12 THEN PRINT " NOT EQUAL" ELSE PRINT
80 NEXT X
```

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

Last month I promised that I would eat some words this month. In the first column I wrote for 68 Micro Journal, I said that I was sorry no one was using more than 64K for a single program under OS-9, and I made the point rather strongly that 6809-based computers should not be shared.

Several months ago Oavid Brown asked me to look at his version of MUMPS for the 6809. Strictly speaking, since MUMPS doesn't run under OS-9, it is out of my area, but it is intriguing. The version of MUMPS David Brown sent me uses a fairly sophisticated virtual memory scheme, and is not effected by 64K boundaries. Since it doesn't run under OS-9, I still challenge someone to be the first with a program that uses more than 64K at once under OS-9, but since Dave Brown's work is impressive, I gave it a mixed but generally nice review.

My mother is the secretary of the school board back in the town where I grew up. She has given me a very interesting pipeline into the workings of a municipal school system. Recently there has been a lot of fuss about computers at school. Pre-college schools have to make a number of difficult decisions in the process of integrating computers in the educational process. Even the choice of the best computer is complicated for them by the scarcity of good software for their purposes (and their uncertainty concerning what software they need), and by the worst kind of financial problems. When I heard that my home town was going to commit itself to a gaggle of microcomputers running Basic, I felt motivated to research the subject with an eye toward talking them out of Basic. The OS-9 users' group's bulletin board is often a good source of information, and in this case it was surprisingly useful; it turns out that many OS-9 users are involved in education. Once started on the idea that OS-9 might be a good solution for some of a schools system's problems, I rubbed some figures together and came to some conclusions that shouldn't have surprised me.

It is clear that financial considerations are crucially important to all the school systems I know of. One micro can be inexpensive enough to fit into a budget, but one Apple is not very useful for teaching a class of thirty. I figure that a high school computer lab should be set up to teach Pascal, word processing, the use of a spread sheet, and the use of computers in the sciences. I know from experience that students can be lab partners and work as a team of two without too much trouble, but three or more students working together will have problems. Figuring thirty students in a class, the lab will need fifteen stations. The minimum configuration I can put together is fifteen micros, each including:

A spread sheet	-- $100.00
Pascal	-- $200.00
Wordstar type editor	-- $300.00
Operating system	-- $100.00
One 5.25" floppy drive and controller	-- $600.00
A printer	-- $250.00
A monitor	-- $200.00
The micro	-- $500.00

All those prices are rough, but reflect the cheap alternative, not the quality that students deserve. Each micro will come to $2250.00 (though I doubt that they could actually be put together for that little). Fifteen of them cost $33750.00. That's serious money, and it only buys a minimal system for each lab team.

If a large OS-9 system could handle fifteen students, it would be possible to purchase a top of the line CPU with a hard disk, a floppy disk, fifteen serial ports (intelligent), a half meg of memory, and top of the line software, for about $14,000.00. Fifteen very nice terminals would cost $9000.00 bringing the cost of the system to $23,000.00. Two thousand dollars will buy a very nice printer, bringing the total cost to about $25,000.

I have talked to several people who run many users on a Gimix-III system. If half of what the Gimix-III users say is true, it would be reasonable to have eight or ten students sharing a machine. If all that they say is true, it might be possible to hook thirty students to one CPU and expect them to run at a reasonable speed. I now have a second terminal on my level two system. I can say from my own experience that my system can handle two users with very few signs of being loaded down.

Based on what I know about my system, and what I have been able to find out about Gimix-III, I think a Gimix-III system with at least 256K of memory would be able to handle four to six users with a level of service that I would find acceptable. Given a choice of a toy computer with bargain basement software, and the bare minimum of peripherals, or a fifteenth of a fully configured Gimix-III system; I would pick the piece of a large system like a flash.

I confess to being an ivory tower idealist. I want people to like computers, so I flinch at the idea of giving out slices of computer so small that there is not enough power to allow software to be friendly. That means that I think a individual deserves at least a level two system with lots of memory. Realistically, most hobby types can't afford to commit that much money to their computer; businesses need a much stronger argument than friendly relationships between staff and computers; and schools simply have to choose the least expensive way to do things most of the time. I maintain that I am philosophically opposed to sharing micros, but if I am forced to consider the alternatives, I am strongly in favor of sharing a computer -- provided it is the right computer.

A Letter

Don Williams sent me a letter from Bengt-Allan Bergvall who sent along an interesting program that amounts to a special sort of shell for Basic09 programs. It gives me encouragement in my plan to write a enhanced shell, but is useful as it stands. What follows is an excerpt from his letter:

==

Parameter List Modifier for Basic09

Microware's BASIC09 is an excellent interpreter, easy to use for producing your own utilities. Unfortunately, it is lacking a straightforward method of passing parameters. For example, if you are going to write a "Help" utility, you want to type

OS9:help dir to learn about the dir command. This is impossible if Help is a BASIC09 program.

If Help is a packed BASIC09 program, interpreted by RunB, you can type

OS9:help only, and let the program ask you what help you want. If you don't have RunB, you have to type

```
OS9:basic09 5k help
```

However, even if Microware doesn't tell you, you can also pass parameters in RunB or BASIC09 by using the syntax

```
OS9:help ("dir")    or OS9:basic09 5k help("dir")
```

and using the PARAM statement in the Help program. This is OK if you will use the program rarely, but if the program will be used often, and perhaps not by yourself, this is a very clumsy syntax.

The desired syntax can of course be accomplished by writing Help in another language that permits the desired parameter syntax, i.e. in assembler. This is probably the wrong way for a user utility program.

To solve the problem, I have written a short "universal" program in assembler, called ParamMod, with the following characteristics: * ParamMod allows the desired parameter syntax. * ParamMod transforms the parameter list from the desired syntax to the syntax required by BASIC09 or RunB. The resulting parameters are all of the type STRING. To be used as numeric types, the strings have to be transformed using the VAL function. * ParamMod forks to either BASIC09 or RunB, and the main program is written in BASIC09. * ParamMod has to duplicated and customized on three text strings and needed BASIC09 memory for each utility:

 Innam: The wanted utility name. In the given case,
 Help. Other utilities could be names Compare or
 Analyze.
 Outname: The name of the file that contains the BASIC09
 procedure and performs the desired action. It
 could be named Help_B or /D0/COM/Compare_B or
 AnalyzeBody.Interprt.
 Interprt: The name of the BASIC09 interpreter to be
 forked to. Either BASIC09 if outname is a saved
 procedure or RunB if it is a packed procedure.
 Memory: The total number of bytes needed for the
 procedures and their data areas.

In the following, we are assuming you are writing a Help utility. For other utilities, change the names accordingly.

First customize ParamMod's three text strings and BASIC09 estimated memory size with your text editor. Then assemble it with Microware's assembler, using the command:

```
OS9:asm ParamMod o=Help 10k
```

and the resulting code for Help will be in your execution directory.

Then write your BASIC09 program, naming the outermost procedure Help_B. You must save or pack Help_B to run it through Help. RUn it from within BASIC09 with the command (including parameter):

```
B:$help dir
```

You may also during the development phase run the program without Help. In that case you must use BASIC09 parameter syntax:

```
B:run help_b("dir")
```

Included is the assembly listing for ParamMod, customized for a Help utility and a dummy Help_B program.

Bengt Allan Bergvall Blavingev. 1 S-561 49 Huskvarna Sweden

==

The assembler program which was included with the letter, and which I will include here, is an interesting extension on the program called "StrtTask" which I gave a few months ago. If we were using real UNIX we would solve the problem of passing parameters to BASIC09 programs by modifying the shell; ParamMod is a sort of special purpose mini-shell which runs BASIC09 programs.

```
         * Program written by Bengt-Allan Bergvall, Blavingev. 1,
         * s-561 49 Huskvarna, sweden.
         *
         * Program to reformat a parameter list from an easily
         * typed form to the clumsier form required when running a
         * BASIC09 program.
         * Given the command
         * OS9:help param1 param2   param4  (note the extra space)
         * This program will fork to the RunB or BASIC09 program
         * Help_B as if given the equivalent command:
         * OS9: BASIC09 #5k Help_B("param1","param2","","param4")
         *
         *
         * This program is general and can reformat the resulting
         * parameter list up to 256 characters, but the name
         * strings inname and outname has to be changed for each
         * implementation.
         *
         * if interprt is runB, then outname has to be a packed
         * BASIC09 program in the execution directory.
         *
         * If interprt is BASIC09 then outname has to be a saved
         * BASIC09 program either in the present data directory or
         * in another file with outname giving the full path name,
         * e.g., /D0/CDM/Help_B
         *
         * The memory needed by BASIC09 or RunB must also be
         * given.
                                  nam   parameter list modifier
                                  ttl   for BASIC09 or RunB
                                  ifp1
                                  endc              (use os9defs)
   0000 87CD0092                  mod   pgend,inname,prgrm+objct,reent+1
   0009 001F01CA                  fdb   pgstart,stack
         * data variables
 D 0000             parend5       rmb   2           output parameter limit -5
 D 0002             outpar        rmb   256
 D 0102             varend        equ   .
 D 0102             stck          rmb   200         stack area
 D 01CA             stack         equ   .           stack pointer
         ****************************************
         ** Customization area

   000D 48656CF0    inname        fcs   /Help/      Name of utility
   0011 48656C70    outname       fcs   /Help_B/    Name of BASIC09 procedure
   0017 00                        fcb   0
   0018 42415349    interprt      fcs   /BASIC09/   Either BASIC09 or runB
         * Total memory needed in bytes by BASIC09 or RunB
         * process: (equivalent to the needed BASIC09 MEM value)
   1388             memory        equ   5000

         ** End customization
         ****************************************
   001F             pgstart

         * Modify parameter list from free form into BASIC09
         * string form. Example of free form: param1 param2  param4
         * Resulting BASIC09 string form:
         *  Help_B("param1","param2","","param4")

         * prepare limit check for parameter list, allow for
         * ending last parenthesis.
   001F 31C900FD                  leay  varend-5,U
   0023 109F00                    sty   parend5

         * copy outname into output parameter list
   0026 3410                      pshs  X
   0028 3142                      leay  outpar,U
   002A 308DFFE3                  leax  outname,PCR
   002E A680        namechar      lda   ,X+
   0030 2704                      beq   nameend
   0032 A7A0                      sta   ,Y+
   0034 20F8                      bra   namechar
   0036 3510        nameend       puls  X           input parameter list

         * append modified input parameter list to output
         * parameter list
   0038 8628                      lda   #'(
   003A A7A0                      sta   ,Y+
   003C 8622                      lda   #'"
   003E A7A0                      sta   ,Y+
   0040 A680        parchar       lda   ,X+
         * check the resulting parameter list not too long
   0042 109C00                    cmpy  parend5
   0045 2506                      blo   parOK
   0047 53                        comb              set carry
   0048 C638                      ldb   #56         BASIC09 parameter error
   004A 103F06                    OS9   F$Exit

   004D 8120        parOK         cmpa  #$20        space?
   004F 2708                      beq   nextpar
   0051 810D                      cmpa  #$0D        carriage return ends parameter
   0053 2712                      beq   lastpar
   0055 A7A0                      sta   ,Y+
   0057 20E7                      bra   parchar
         * reformat next parameter
   0059 8622        nextpar       lda   #'"
   005B A7A0                      sta   ,Y+
   005D 862C                      lda   #',
   005F A7A0                      sta   ,Y+
   0061 8622                      lda   #'"
   0063 A7A0                      sta   ,Y+
   0065 20D9                      bra   parchar
         * list end
   0067 8622        lastpar       lda   #'"
   0069 A7A0                      sta   ,Y+
   006B 8629                      lda   #')
   006D A7A0                      sta   ,Y+
   006F 860D                      lda   #$0D        carriage return
   0071 A7A0                      sta   ,Y+

         * fork to interprt (RunB or BASIC09)
   0073 308DFFA1                  leax  interprt,PCR
   0077 108E0100                  ldy   #$100       allow one page parameters
   007B 3342                      leau  outpar,U
   007D 8611                      lda   #prgrm+objct
   007F C614                      ldb   #(memory+255)/256 data area
   0081 103F03                    OS9   F$Fork
   0084 2508                      bcs   ut
   0086 103F04                    OS9   F$Wait
   0089 2501                      bcs   ut
   008B 5F                        clrb              no error
   008C 103F06      ut            OS9   F$Exit
   008F 1AAD9C                    emod
   0092             pgend         equ   *
                                  end

00000 error(s)
00000 warning(s)
$0092 00146 program bytes generated
$01CA 00458 data bytes allocated
$2104 08452 bytes used for symbols
```

```
PROCEDURE Help_B
REM Dummy Help utility
REM prints the parameter
PARAM text:STRING
PRINT text
BYE  \REM bye needed to give automatic return to OS-9 when run by Basic09
```

```
* Program written by Bengt-Allan Bergvall, Blavingev. 1,
* s-561 49 Huskvarna, sweden.
*
* Program to reformat a parameter list from an easily
* typed form to the clumsier form required when running a
* BASIC09 program.
* Given the command
* OS9:help param1 param2   param4  (note the extra space)
* This program will fork to the RunB or BASIC09 program
* Help_B as if given the equivalent command:
* OS9: BASIC09 #5k Help_B("param1","param2","","param4")
*
*
* This program is general and can reformat the resulting
* parameter list up to 256 characters, but the name
* strings inname and outname has to be changed for each
* implementation.
*
```

```
* if interprt is runB, then outname has to be a packed
* BASIC09 program in the execution directory.
*
* If interprt is BASIC09 then outname has to be a saved
* BASIC09 program either in the present data directory or
* in another file with outname giving the full path name,
* e.g., /D0/COM/Help_B
*
* The memory needed by BASIC09 or RunB must also be
* given.
 nam parameter list modifier
 ttl for BASIC09 or RunB
 ifpl
 use /D0/DEFS/defslist
 endc (use os9defs)
 mod pgend,inname,prgrm+objct,reent+1
 fdb pgstart,stack
* data variables
parend5 rmb 2 output parameter limit -5
outpar rmb 256
varend equ .
stck rmb 200 stack area
stack equ . stack pointer
****************************************
** Customization area

inname fcs .Help. Name of utility
outname fcs .Help_B. Name of BASIC09 procedure
 fcb 0
interprt fcs .BASIC09. Either BASIC09 or runB
* Total memory needed in bytes by BASIC09 or RunB
* process: (equivalent to the needed BASIC09 MEM value)
memory equ 5000

** End customization
****************************************
pgstart

* Modify parameter list from free form into BASIC09
* string form. Example of free form: param1 param2  param4
* Resulting BASIC09 string form:
*  Help_B("param1","param2","","param4")

* prepare limit check for parameter list, allow for
* ending last parenthesis.
 leay varend-5,U
 sty parend5

* copy outname into output parameter list
 pshs X
 leay outpar,U
 leax outname,PCR
namechar lda ,X+
 beq nameend
 sta ,Y+
 bra namechar
nameend puls X input parameter list

* append modified input parameter list to output
* parameter list
 lda #'(
 sta ,Y+
 lda #'"
 sta ,Y+
parchar lda ,X+
* check the resulting parameter list not too long
 cmpy parend5
 blo parOK
 comb set carry
 ldb #56 BASIC09 parameter error
 OS9 F$Exit

parOK cmpa.#$20 space?
 beq nextpar
 cmpa #$0D carriage return ends parameter list
 beq lastpar
 sta ,Y+
 bra parchar
* reformat next parameter
nextpar lda #'"
 sta ,Y+
 lda #',
 sta ,Y+
 lda #'"
 sta ,Y+
 bra parchar
* list end
lastpar lda #'"
 sta ,Y+
 lda #')
 sta ,Y+
 lda #$0D carriage return
 sta ,Y+

* fork to interprt (RUnB or BASIC09)
 leax interprt,PCR
 ldy #$100 allow one page parameters
 leau outpar,U
 lda #prgrm+objct
 ldb #(memory+255)/256 data area
 OS9 F$Fork
 bcs ut
 OS9 F$Wait
 bcs ut
 clrb no error
ut OS9 F$Exit
 emod
pgend equ *
 end
```

OS9 2nd Annual Seminar

Microware Corporation, the developers of OS-9 held their second annual OS-9 Seminar in Des Moines, Iowa the 12th, 13th, 14th and 15th August '83. To report the entire affair in detail would probably take up this entire Issue. So I will attempt to cover what I thought were the highlights and happenings that you might find interesting.

There were over 400 (estimate) users attending and a total of 23 exhibitors showing their OS-9 hardware and software. The exhibitors were:

OS-9 Users Group
Positron Ltd.
JBM Group Inc.
Gimix Inc.
Micro Intl.
Microware Systems Corp.
Specialty Electronics
Great Plains Computers
Applied Computer Tech.
Hazelwood Computers
Advanced Digital Tech.
Privac, Inc.
Motorola
Ackerman Digital Inc.
Forum 68
Tano Corp.
FHL
Candid Logic
Smoke Signal Broadcasting
Computer Sys Consultants
Computer Systems Center
Matrix Corp.
Creative Micro Systems

My wife Joyce and I arrived early Friday morning having decided to drive (930 miles) the night before as the weather was forecast to be too bad to fly our light plane to Des Moines. Needless to say it was clear as a bell all the way, oh well. Was met at the door of the Marlott Hotel by Richard Don of Gimix who was apparently looking out for all his 'chicks'. This was about 1:30 a.m. Friday morning. After exchanging some (can't remember what) I staggered off to bed after a straight 19 hours of driving. So ended day one.

Friday was sort of a hectic set up day for the exhibitors as the attendance was expected to be greater than originally anticipated. It was. I was personally surprised at the intensity and quality of the hardware and software shown. OS-9 has sorta 'crept' up on the S50 Bus community. Yes, it has been here for over 4 years, but we, or at least I, did not actually comprehend just how solid the growth of OS-9 has been. In my conversations with many others, I gather that they are also somewhat surprised by the depth and penetration made by OS-9 in the past few years. OS-9 has injected new vigor and interest in the 6809 and Standard S50 Bus computers in general, and that can only speak well to the future.

Friday night, the 12th, we gathered for a welcoming banquet at the hotel. The guest speaker was no less than the governor of the state of Iowa, Jerry Branstad. Now I have been to a lot of computer conventions and such but this is the first time that I can remember that the governor came, with his wife, and addressed the group. Fact is he indicated that he and Ken Kaplan have a lot in common, they both are expecting new additions to the family shortly.

I would imagine that by the time this is delivered to you, young Master Aaron Kaplan will have arrived. I sorta stood around all weekend, with a stock of my best cigars, waiting for his arrival. I figured Ken might need some help from an experienced 'cigar hander outer'. But as of this date, nearly a week later, Aaron has refused to cooperate. He is past due and I think I know the reason; he just wants to make sure that Dad and all the others down at Microware have all the major bugs out of OS-9. Well, I can tell you son, from what I saw, have no fear, things sure seemed to be running smooth, and I talked to a lot of very satisfied users from all over the world.

Also during the banquet Ken Kaplan rose and announced that there were some awards to be presented. The following is the 'scrolls' from which the words of praise and predictions were delivered to all. A nice trophy was presented to all honored by this prestigious event. I personally was deeply humbled to realize that my goal and burning desire of enlightenment was finally being acknowledged for what it actually has been these long years. I know that Columbus (the one from across the water and not the one in Ohio, Georgia, etc.) now sleeps more securely. 6 out of 10 ain't bad!

THE ALEXANDER GRAHAM BELL AWARD

Richard Don, Gimix, Inc.

Besides being a prominent computer manufacturer, Mr. Don is an outstanding spokesman in our industry. Outstanding indeed! His unique method of direct communication with a wide audience every night by telephone is truely awesome. Very few Americans appreciate the fact that we would not have the world's finest telephone system if Richard Don were not around to pay for it. Even fewer people are aware of the fact that the upcoming breakup of AT&T is the direct result of one weekend last winter when Richard had laryngitis. We are confident that Mr. Don's tireless efforts will someday help us achieve the goal of replacing every audio telephone set in the country with auto-answer modems. Mr. Don, we salute you!

THE STIMULATION IN ADVERTISING AWARD

Frank Hogg, Frank Hogg Laboratories.

Mr. Hogg is well known as a major and innovative 6809 software distributor. By running his now-famous ad of an attractive young lady clad only in a nightgown, Frank Hogg Laboratories has advanced the art of software marketing from its primitive tasteful status quo to the more stimulating and sophisticated techniques employed by big Madison Avenue agencies to sell products such as razor blades, motorcycles, waterbeds, and lite beer. We understand that after this ad hit the streets the phones at Frank Hogg Labs rang off the hook for weeks but unfortunately most callers wanted the lady's phone number instead of software. But don't worry Frank, if it works for Gillette it will work for you too. Unfortunately, Mr. Hogg is not here to accept his awards as he has a date with you-know-who, so accepting the award in his place will be Frank Hoffman.

THE CRAY 2 AWARD

Ric Hammond, Smoke Signal Broadcasting

It would have never occurred to us sleepy backwater folks that OS-9 is the ideal operating system for the CRAY-2 supercomputer. Had it not been for Mr. Hammond's suggestion and encouragement Microware probably would never have done the CRAY-2 version of OS-9. Even though he didn't really think of it until Smoke Signal's marketing manager, Jim Lilley, had already sold three copies, we feel he still deserves this award. By the way, those of you with Cray 2s at home can get your own copy as soon as Smoke Signal finishes up the joystick drivers. Mr. Hammond, we salute you!

THE SYSTEM SCREAM AWARD

Mr. Jack DeFrial, Advanced Digital Technology

Mr. DeFrial is a prophet of a magic box called a real-time emulator system. This ingenious device has contributed greatly to the mental health of 6809 programmers by regularly delivering irrefutable evidence that most bugs are really hardware problems. Upon offering the appropriate incantations to this magic box, it will transform an ordinary dumb terminal into a crystal ball which allows one to peer into the mysterious time elements of microseconds and nanoseconds. It is within this strange world that mischevious logic circuits play havoc upon even the most highly structured code. This is something we programmers have suspected for a long time but were unable to prove until this amazing device came along. Mr. DeFrial, we salute you!

THE BARTER ECONOMICS AWARD

Mr. Pete Prossen, Creative Micro Systems

Pete Prossen is one of our favorite people. His company makes top notch Exorbus systems and boards and he was one of the very first 6809 manufacturers to take a chance on an unproven operating system called OS-9. While his company and products are quite well-known, one little known fact about Pete is that he also owns an avocado grove. We really like avocados but unfortunately they are not indigenous to this area. So we made an interesting deal - we ship software to Pete and he ships avocados back to us. This is a classic example of an advanced economic concept called "Barter Trade", and we think its great. The only real problem with it is that we have not been able to get the tax collector to accept vegetables yet. In case some of you who are our customers are thinking of similar schemes, we must warn you that we already have plenty of disks, rams, telephones, computers, automobiles, assorted control systems, software, cotton, corn, rice, and soybeans. However, we are still in need of gold, silver, platinum, diamonds, rare coins, or original 15th century old masters. For being a pioneer in barter economics. Mr. Prossen, we salute you!

THE INTERNATIONAL GEOGRAPHIC AWARD

Mr. Don Williams, 68 Micro Journal

A recent survey indicated that over 6 out of 10 6809 programmers now believe the earth is indeed round. For this amazing feat of mass education we must give full credit to Mr. Don Williams, publisher of 68 Micro Journal Magazine. Since its inception, the cover of every issue of 68 Micro Journal has had a full color satellite photograph of the planet Earth which clearly exhibits its round shape. We also wish to refute the vicious rumor that the reason Mr. Williams has run the same cover for the last five years is that he got a hell of a deal on two and a half million unused covers for the now-defunct magazine, "Albanian Space Research Digest". But we do not know if there is any truth behind the rumor that over the next five years 68 Micro Journal will switch its cover to a photo of a microprocessor chip in order to teach programmers that integrated circuits are rectangular objects. Mr. Williams, we salute you!

Saturday the 13th, started off at 8:00 a.m. with an audio-visual presentation to Microware. followed by Indepth classes and demonstrations of Microware software, and languages running under the OS-9 disk system. The entire crew from Microware, including Ken Kaplan attended these sessions and fielded any and all questions, suggestions and comments from the large group gathered. At times several hundred would be gathered, and each one who desired to be heard was acknowledged. The exchanges were were both informative and constructive to all who attended and expressed their impressions to me.

The topics covered during that day were, of course, OS-9 and Basic09. Additionally an OS-9 roundtable, a free discussion period lasting over 2 hours, was heavy in attendance with all getting a chance to say their piece. I attended this session and do not remember any serious negative comments, from any participant. Frankly I was a bit surprised, I have seldom seen this close a working relation between vendor and users before. Everyone had a chance to 'let the hair down', but I saw little if any dandruff. This speaks well for the apparent excellent relationship between Microware and their customers.

The days sessions ended (but not the days activities) with a general language roundtable, again attended by almost all of the Microware crew, including Ken. At this time Microware introduced James McCosh, author of Microware's 'C', also Keith Ashman discussing a powerful developement language 'Sage'. Both had come from England to attend and field questions and comments concerning their products. By this time the prearranged schedule had 'gone-to-pot' as the OS-9 Users Group who had intended to meet at 5:30 p.m. finally got started sometime after 8:00 p.m., if my memory serves me correct.

Saturday evening the 68 Micro Journal banquet got underway about 5:30 or so. This was a wonderful opportunity for me to say thanks to many there who have supported 68 Micro Journal over the years. I had no intention of giving a pep or sales pitch, just a chance for a lot of old friends to get together. Over 75 gathered and as we were getting started Ken Kaplan and Richard Don (Gimix) 'captured' the groups attention by pounding silverware on glasses (seems like it is hard sometimes to get the attention of a group of Standard S50 Busers). Well, to say the **VERY LEAST** I got a real **SURPRISE**. There I sat, patting myself on the back for not imposing a 'pitch' on my friends, when these two took control. "What", I thought, "some folks just never let an opportunity pass." About that time the doors opened and in walked a serving crew with a candle lit birthday cake. Ken and Richard led the crowd in singing "Happy Birthday - Dear Don". All I can ever say is thanks Jeanne, Ken, Arlene, Richard and all the others who were in on this well kept secret! The 13th was my birthday, won't say how many. For those of you who know me, it was a rare moment of speechlessness. **I was really dumbfounded!!** I want to thank everyone for making that one of the nicest moments in my entire life. Despite the fact that not one voice in the crowd seemed to be capable of carrying a tune, yet how sweet it was.

Saturday night extended well into the morning as Joyce and I attended the hospitality suites of MICROWARE, GIMIX, SMOKE SIGNAL BROADCASTING, ACKERMAN DIGITAL and ADVANCED TECHNOLOGY. Regretfully we missed the HAZELWOOD hospitality suite as they were gone to some of others when we called. So went the day.

Sunday started a little late with a brunch hosted by Microware. Afterwards the conference room filled to near capacity as Microware introduced their new OS-9/68000. From the response voiced by potential license prospects and users, the future looks bright. Especially for those who have existing hardware and OS-9 software presently running. It appears that the 'carry-over' from 6809 OS-9 and 68000 OS-9 will be about as painless as it could be done. The 'basic language' (no pun intended) will be 'C'. However, it appears that most all the other high level language source programs will be portable, including Basic09. That means that the present users do not have to hold off their applications software developement, they will be able to port it across to the 68000. To me that is one of the more important aspects facing the user who wishes to upgrade. The time and expense can be fatal, in some cases, otherwise.

Following this session additional discussions were held on the new OS-9 'C'. The 'C' language seems to be making fast inroads in the 6809 community. I suspect it will be the predominant language a couple years hence.

Additional new Microware OS-9 software was presented, including a new relocatable macro assembler, extended utilities and a level II print spooling system.

Later Sunday afternoon presentations were made on behalf of the 'SAGE Applications Generator System', CANDID LOGIC INC.'s, MICRO INTERNATIONAL, EUROMAK COMPUTER SYSTEMS, KEMTRONIX LTD, LEARNING VENTURES, UNISOFT (TMP) and Brian Capouch's programs and systems. This went on until the hall closed about 6:00 p.m.

Monday morning the Microware crew reassembled and again individual questions and problems were discussed with many (few had departed) attending. At no time did it ever appear that Microware failed to meet head-on and reply to all questions, comments and problems. This was commented on many times by users I talked to. They were both impressed and grateful that Microware took sufficient time to listen and respond in depth to all.

Monday afternoon was spent by all being invited to the Microware Works for a tour of the facilities, which Microware will be vacating soon for much larger and newer quarters. As for the 'boss' and I we piled in the heap and headed home to the hills of Tennessee. The drive back was pleasant and enjoyable, despite the 19 hours of steady driving. All in all it was a very pleasant and educational four days.

OS-9 and Radio Shack

While we were at the Microware shindig, Tandy-Radio Shack released the official word (via mailed catalog) of the availability of Microware's OS-9 disk system for the present Color Computer. It will sell for $69.95 on disk with one of the better 'how to' manuals on OS-9. It comes complete, at this price, with editor, assembler and debugger. The standard, full feature Microware Basic09 is also available for $99.95. Dale Puckett, newly elected president of the OS-9 Users Group, has written a very informative book on how to use Basic09 that will be available soon.

While on the subject of the OS-9 Users Group, our very own Peter Dibble was elected vice-president. Congratulations to Dale, Peter and all the other incoming officers. Also our thanks to Brian Capouch and his officers for their efforts this past year on behalf of the OS-9 Users Group. The clo e cooperation between Microware and the Users Group can only strengthen the bonds already established.

The decision by Radio Shack to adopt the Microware S-9 disk system for their products, speaks well of things to come. With nearly a half million of color computers out there, and all now capable of running a full blown multi-user, multi-tasking disk system literally blows the mind. The potential for the software and add-on market is counted in the millions, real spendable dollars that is. Like I said back when most everyone, including Radio Shack was derating the color computer, it had potential that even the designers overlooked. Manufacturers and publishers alike, who just a few months ago were belittling the CoCo, now are madly scrambling to climb aboard. Microware picked a plump plum there!!! And we were telling you about the latent power just waiting to be exposed, since day one.

Also while at the seminar I was gratified with the response our new COLOR MICRO JOURNAL has received. Seems that our basic goals carry over to Color Micro Journal also. We never have pretended to be a 'slick'. Fact is we went from slick paper to a duller matted paper because too many complained about the glare from slick paper while attempting to copy one of the many source listings from 68 Micro Journal. Also we have published more articles, words or what have you, concerning the Standard S50 Bus and the 68XX series of computer devices, **than all the other magazines combined!** And we intend to continue to lead. Maybe we won't have a lot of cartoons, space wasting headers and all the other ploys that publishers and editors do to grab your attention. Never had them and never will (as the ad goes). What we have had and will continue to have is gut level information that allows you, our readers, to gain the most from your 68XX computer. At times some of it is small or hard to read, but we have so much to try to offer you that other magazines would just not mess with, we print it. Knowing that it is valuable to many of you and if we didn't print it - no one would. Far better than 'file 13'. So will be the format of COLOR MICRO JOURNAL also.

I am convinced of this more strongly after our initial release of issue one this past month. We reached more than 30,000 - RIGHT - more than 30,000 readers with the first issue. And now with our background and firm footing in the 6809 world, COLOR MICRO JOURNAL will soon be a leading force in specialty publishing.

An important aspect is that many users will find that there are some built-in limitations in the Color Computer, but having invested time and precious dollars in a Color Computer and OS-9 software, they will, by necessity, want to upgrade to more expandable 6809 computers. That alone will expand the potential market of our present advertisers many thousand fold, especially those catering to the OS-9 marketplace. Back when other publishers ignored the 68XX market, we kept the faith, along with all of you. As the community prospers, in the future, the spillover will reward us all. You will have a choice of more and better products. A large support and user base makes for better and less costly goods. Our market is no different. And our experience and qualifications will enhance both 68 MICRO JOURNAL and COLOR MICRO JOURNAL. Which means that with time we can serve you better. And that has been my goal since we first published, more than five years ago.

Left to right: The Honorable Jerry Branstad, Governor of the State of Iowa, Ken Kaplan, founder and President of Microware Corp., originators of OS9-OS9/68000 systems.

Continued on Page 46

OS9 File Sort

File Sorting for the OS9
by
John J. Strunk

I have been dabbling in computers of one sort or another for about fifteen years and during that time I have found that one of the most indispensible software items is a good file sorting program. File sorting is an integral part of applications such as Mailing List generators, Accounting packages, general data file managers, etc. The list goes on and on.

When I received my new OS9 operating system from Microware Inc., I began to convert my programs to operate with it. Then I realized that no file sorting programs were available for the OS9. What a surprise! The decision then was to either forget using OS9 or generate my own file sorting program. Since OS9 has such vast potential I decided on generating the necessary sorting software. What follows are my efforts to this end.

The SORTFILE program provides the means to sort fixed length files, producing either the sorted records, sorted fully concatenated key, or both. Files to be sorted may contain up to 65535 records, where each record may be up to 1024 bytes in length. There are no restrictions on the contents of these records. The fully concatenated key may contain up to seven independently specified subkeys as long as the total length does not exceed 253 bytes.

SORTFILE was designed to run under the OS9 Level I operating system. However, I know of no reason why it would not operate successfully under the OS9 Level II system as well. Any hardware configuration which supports the OS9 operating system and also has at least one disk drive (any type) is all that is necessary.

The code and data areas for SORTFILE take less than 3K of RAM and the module is fully relocatable and reentrant. Since SORTFILE uses disk space for storing keys and pointers it is never necessary to use additional RAM.

The sorting parameters for SORTFILE can be entered in two different ways. First, these parameters can be stored in a CONTROL_FILE (See Example 1) and this file can then be referenced in the input line for SORTFILE.

An example would be:

```
     SORTFILE CONTROL_FILE
```

Second, these parameters can be entered via the terminal (See Example 2). This is accomplished by entering the SORTFILE command without a CONTROL_FILE specification following it. the program will then prompt you for the needed parameters. An example would be:

```
     SORTFILE
```

When the parameters are entered via the terminal you cannot use OS9 output redirection. This is because the prompts will also be redirected but the input is still expected from the terminal.

The SORTIN_FILE_SPECIFICATION, SORTOUT_FILE_SPECIFICATION, and SORT_WORK_FILE_SPECIFICATION are standard OS9 file names. Their names must conform to the standard OS9 naming conventions. Further, they must be resident on a RBF supported device. All disk drives are RBF devices. The SORT_WORK_FILE_SPECIFICATION and SORTOUT_FILE_SPECIFICATION must not define files existing prior to running SORTFILE.

The DISPOSITION field describes the action to be taken with respect to its associated file. A disposition of KEEP or K will cause the file to be saved after completing the sort operation. A disposition of DELETE or D will cause the file to be deleted after the sort operation. The default dispositions are KEEP for both the SORTIN_FILE and SORTOUT_FILE and DELETE for the SORT_WORK_FILE.

If a disposition of Delete is specified for the SORTOUT_FILE_SPECIFICATION, this file will never be created. Thus, you can specify either a dummy path description or no path description at all! However, if none is specified, the comma must still precede the disposition parameter. Examples are:

```
     DUMMY,Delete
     ,Delete
```

The RECORD_LENGTH field specifies the total length of the record to be sorted. It must include any end-of-field and end-of-record bytes used in the file. This value must be in the range of 1 to 1024. Any other values or any non-numeric entries will result in an error.

The KEY1_POSITION field specifies the starting position of the key within the record relative to the beginning of the record. This value must be in the range of 1 to 1024. Also, this value plus the KEY1_LENGTH must be equal to or less than the RECORD_LENGTH. Any other values or any non-numeric entries will result in an error.

The KEY1_LENGTH field specifies the length of the key. This value must be in the range of 1 to 253. Leading zeros are not required. Any other values or any non-numeric entries will result in an error.

The KEY1_SORT_SEQUENCE field specifies whether the sorting on this key is to be in ASCENDING or DESCENDING sequence. The default is ASCENDING sequence. An A may be entered instead of ASCENDING and a D may be entered instead of DESCENDING.

The KEY1_TYPE field specifies the type of field to be sorted. Currently only Binary and Character field sorts are possible. The Binary option causes the field to be sorted in strict binary order. With this type of sort all characters are sorted based upon their binary value. Thus, a capital 'Z' (with a hex value of 5A) would be sorted ahead of a lower case 'a' (with a hex value of 61) in an ascending sort. This sort type is normally used for integer fields. The Character option causes the field to be sorted such that lower case letters are integrated with the upper case letters. Thus, an upper case 'A' will be sorted together with a lower case 'a'. This sort type is used for character fields and is the default type.

The first key data set specifies the primary sort and must always be present. The second key data set specifies the secondary sort and so on. The second and subsequent key data sets are optional.

The DISPOSITION, KEY1_SORT_SEQUENCE, KEY1_TYPE, and END (input delimiter) may be upper and/or lower case characters.

All errors reported by SORTFILE are standard OS9 defined error codes. The most common error codes are listed in Table 1.

When the sorting operation is complete a statistics report will be produced on the standard output device. See Example 3 for a sample of this report.

There are no restrictions on using my SORTFILE program for personal use. If you do use SORTFILE and have problems with it or you have comments or suggestions please write me a note.

```
SORTIN_FILE_SPECIFICATION,DISPOSITION
SORT_WORK_FILE_SPECIFICATION,DISPOSITION
SORTOUT_FILE_SPECIFICATION,DISPOSITION
RECORD_LENGTH
     KEY1_POSITION,KEY1_LENGTH,KEY1_SORT_SEQUENCE,KEY1_TYPE
     (Up to 6 additional sets of key specifications)
     END        (Input delimiter)
```

Example 1. Typical CONTROL_FILE format.

```
SORTIN FILE =======> SORTIN_FILE_SPECIFICATION,DISPOSITION
SORTWORK FILE =====> SORT_WORK_FILE_SPECIFICATION,DISPOSITION
SORTOUT FILE ======> SORTOUT_FILE_SPECIFICATION,DISPOSITION
RECORD LENGTH =====> RECORD_LENGTH
SORT KEYS (POSITION,LENGTH,SEQUENCE,TYPE)
     KEY1_POSITION,KEY1_LENGTH,KEY1_SORT_SEQUENCE,KEY1_TYPE
     (up to 6 additional keys may be specified)
     END      (Input delimiter)
```

Example 2. Typical input via terminal. The fields in italics are operator inputs.

```
SORTFILE  (VERSION 1.2)
Copyright 1981 by John J. Strunk

00080 RECORDS SORTED

SORTIN FILE =======> /D3/ADDRESSES,KEEP

SORTWORK FILE =====> /D3/SORT_WORK_FILE,DELETE

SORTOUT FILE ======> /D3/STATE_SORT,KEEP


RECORD LENGTH =====> 00210

SORT KEYS (POSITION,LENGTH,SEQUENCE,TYPE)
00121,00015,ASCENDING,CHARACTER
00151,00005,ASCENDING,CHARACTER
END
```

Example 3. Typical statistics output report.

Table 1

Common Error Codes with Reasons

Error Code	Error Description	Reason
16	Illegal Number	1) Either the record length, key position, or key length field contains non-numeric characters.
		2) Either the record length, key position, or key length field is missing or is zero.
77	Illegal Input Variable	1) The record length specified is greater than 1024 bytes.
		2) The key position field specifies a value beyond the limits of the record.
		3) The sum of the key position value and the key length extends beyond the limits of the record.
		4) The sum of all the key lengths is greater than 253 bytes.
90	Control input missing	1) The record length, key position, or key length field is missing.
		2) The END statement (input delimiter) is missing.
211	End of File	1) The record length specified for SORTIN_FILE_SPECIFICATION is incorrect.
216	Path Name Not Found	1) SORTIN_FILE_SPECIFICATION was not found.
218	File Already Exist	1) SORT_WORK_FILE_SPECIFICATION or SORTOUT_FILE_SPECIFICATION already exist.

```
            **************************************************************
            ***** SORT fixed length records module                   *****
            *****                                                    *****
            ***** This module can sort files either in ascending or descending *****
            ***** order up to 65535 records in length. Records may be up to *****
            ***** 1024 bytes in length and the fully concatenated key field *****
            ***** may be up to 253 bytes in length. Up to seven sort keys can *****
            ***** be located anywhere in the record and can be in any order. *****
            *****                                                    *****
            ***** Three files are used during the sort operation along with *****
            ***** the CONTROL_FILE which specifies the sort parameters. These *****
            ***** parameters include:                                *****
            *****       o  Record length                             *****
            *****       o  Sort keys                                 *****
            *****          o  Key position                           *****
            *****          o  Key length                             *****
            *****          o  Key sort sequence                      *****
            *****          o  Key type                               *****
            *****                                                    *****
            ***** All file names follow OS9 conventions.             *****
            *****                                                    *****
            ***** All error codes are standard OS9 codes.            *****
            *****                                                    *****
            ***** Example of CONTROL_FILE:                           *****
            *****                                                    *****
            *****   /D2/SORTIN_FILE,KEEP                             *****
            *****   /D1/SORT_WORK_FILE,DELETE                        *****
            *****   /D2/SORTOUT_FILE,KEEP                            *****
            *****   210                      (Record length)         *****
            *****   121,015,A,C              (Key 1 data)            *****
            *****   0151,5,D,C               (Key 2 data)            *****
            *****   END                      (Input delimiter)       *****
            *****                                                    *****
            ***** The output cannot be redirected with the OS9 > command *****
            ***** unless the control input comes from a file instead of the *****
            ***** terminal.                                          *****
            *****                                                    *****
            *****                   Written 12/26/81 (VERSION 1.0)   *****
            *****                   Revised 01/15/82 (VERSION 1.1)   *****
            *****                   Revised 01/29/82 (VERSION 1.2)   *****
            *****                          by                        *****
            *****                    John J. Strunk                  *****
            *****                    P.O. Box 2242                   *****
            *****                    Richardson, Texas 75080         *****
            *****                                                    *****
            **************************************************************

            *
            * Set OS-9 header variables
            *
      0010           PGM     SET   %00010000  Program module identification
      0001           OBJ     SET   %00000001  6809 object code identification
      0080           REENT   SET   %10000000  Reentrant code identification
      0002           REV     SET   2          Revision level
            *
            * Set OS-9 call codes
            *
      0006           F$EXIT  SET   $06        OS-9 exit routine
      0083           I$CREA  SET   $83        OS-9 create a new file
      0084           I$OPEN  SET   $84        OS-9 open a file
      0087           I$DLET  SET   $87        OS-9 delete a file
      0088           I$SEEK  SET   $88        OS-9 position file pointer routine
      0089           I$READ  SET   $89        OS-9 read a file
      008A           I$WRIT  SET   $8A        OS-9 write a file
      008B           I$RDLN  SET   $8B        OS-9 read line routine
      008C           I$WRLN  SET   $8C        OS-9 write line routine
      008F           I$CLOS  SET   $8F        OS-9 close routine
            *
            * Set up OS9 module header
            *
                             OPT   F,S,W100,D55,0
                             NAM   RECORD SORTING ROUTINE
      0000 87CD0932          MOD   HEEND,NAME,PGM+OBJ,REENT+REV,ENTRY,STOR
      0004 00181102
      0008 55010009
      000C 00                            MOD HEEND,NAME,PGM+OBJ,REENT+REV,ENTRY,STOR
      000D 534F5254  TITLE   FCC   "SORTFILE  (VERSION 1.2)"
      0011 46494C45
      0015 20202856
      0019 45525349
      001D 4F4E2031
      0021 2E3229                        TITLE FCC "SORTFILE  (VERSION 1.2)"
      0024 0D0A              FDB   $0D0A
      0026 436F7079          FCC   "Copyright 1981 by John J. Strunk"
      002A 72696768
      002E 74203139
      0032 38312062
      0036 79204A6F
      003A 686E204A
      003E 2E205374
      0042 72756E6B                          FCC "Copyright 1981 by John J. Strunk"
      0046 0D0A              FDB   $0D0A
      0048 534F5254  NAME    FCS   "SORTFILE"
      004C 46494CC5                      NAME FCS "SORTFILE"
      0050 20524543  RS01    FCC   " RECORDS SORTED"
      0054 4F524453
```

```
0058 20534F52
005C 544544                                      MSGL  FCC ' RECORDS SORTED'
005F 000A          CRLF    FDB   $000A
0061 0C00          HP      FDB   $0C00
0063 534F5254      MSTR    FCC   'SORTIN FILE =====> '
0067 494E2046
006B 494C4520
006F 3D3D3D3D
0073 3D3D3D3E
0077 20                                          MSTR FCC  'SORTIN FILE =====> '
0078 534F5254      OUTP    FCC   'SORTOUT FILE =====> '
007C 4F555420
0080 46494C45
0084 203D3D3D
0088 3D3D3D3E
008C 20                                          OUTP FCC  'SORTOUT FILE =====> '
008D 534F5254      SWORK   FCC   'SORTWORK FILE ====> '
0091 574F524B
0095 2046494C
0099 45203D3D
009D 3D3D3D3E
00A1 20                                          SWORK FCC  'SORTWORK FILE ====> '
00A2 5245434F      RL      FCC   'RECORD LENGTH =====> '
00A6 5244204C
00AA 454E4754
00AE 48203D3D
00B2 3D3D3D3E
00B6 20                                          RL    FCC  'RECORD LENGTH =====> '
00B7 534F5254      SK      FCC   'SORT KEYS (POSITION,LENGTH,SEQUENCE,TYPE)'
00BB 204B4559
00BF 53202850
00C3 4F534954
00C7 494F4E2C
00CB 4C454E47
00CF 54482C53
00D3 45515545
00D7 4E43452C
00DB 54595045
00DF 29                                          SK    FCC 'SORT KEYS (POSITION,LENGTH,SEQUENCE,TYP
00E0 41534345      ASCEND  FCC   'ASCENDING,'
00E4 4E44494E
00E8 472C                                        ASCEND FCC  'ASCENDING,'
00EA 00                    FCB   $00
00EB 44455343      DESEND  FCC   'DESCENDING,'
00EF 454E4449
00F3 4E472C                                      DESEND FCC  'DESCENDING,'
00F6 43484152      CTYPE   FCC   'CHARACTER'
00FA 41435445
00FE 52                                          CTYPE  FCC 'CHARACTER'
00FF 42494E41      BTYPE   FCC   'BINARY   '
0103 52592020
0107 20                                          BTYPE  FCC 'BINARY   '
0108 454E44        ENDMSG  FCC   'END'

     0400          RECMAX  EQU   1024     Maximum record length
     0100          KLMAX   EQU   256      Maximum total key length
   D 0000          R1      RMB   2        Sort work area
   D 0002          CR      RMB   1        Carrage return code
   D 0003          M1      RMB   2        Multiply work area
   D 0005          RESULT  RMB   2        Receiving field for ASCII to binary conversion
   D 0007          COUNT   RMB   1        ASCII to binary digit counter
   D 0008          SEQ     RMB   2        SORTOUT_FILE record counter
   D 000A          ASCII   RMB   5        Receiving field for binary to ASCII conversion
   D 000F          COMMA   RMB   1        Separator
   D 0010          SKEYS   RMB   40       Sort keys storage area (7 keys & one null key)
   D 0038          LRECL   RMB   2        SORTIN_FILE & SORTOUT_FILE record length
   D 003A          IRECL   RMB   2        SORT_WORK_FILE record length
   D 003C          LKEYP   RMB   2        Address of current key position
   D 003E          LKEY    RMB   2        Address of current key length
   D 0040          SORTSQ  RMB   2        Address of current sort sequence
   D 0042          TYPE    RMB   2        Address of current sort type
   D 0044          KEYS    RMB   2        Address of current position in sort keys storage a
   D 0046          INDEX   RMB   2        Address of SORT_WORK_FILE path name
   D 0048          MASTER  RMB   2        Address of SORTIN_FILE path name
   D 004A          OUTPUT  RMB   2        Address of SORTOUT_FILE path name
   D 004C          TSORT   RMB   1        Temporary sort type save area
   D 004D          COMPSY  RMB   1        Compare work area
   D 004E          JPATH   RMB   1        CONTROL_FILE path number
   D 004F          IPATH   RMB   1        SORT_WORK_FILE path number
   D 0050          MPATH   RMB   1        SORTIN_FILE path number
   D 0051          SPATH   RMB   1        SORTOUT_FILE path number
   D 0052          G1      RMB   2        Sort work counter
   D 0054          I       RMB   2        Sort work counter
   D 0056          J       RMB   2        Sort work counter
   D 0058          J1      RMB   2        Sort work counter
   D 005A          POS     RMB   7        Multiply operation product area
   D 0061          REC     RMB   2        Sort record pointer
   D 0063          RECP    RMB   2        Sort record pointer
   D 0065          BUFF    RMB   80        Control input buffer
   D 00B5          MPBUFF  RMB   80       SORTIN_FILE path name
   D 0105          ISBUFF  RMB   80       SORT_WORK_FILE path name
   D 0155          OTBUFF  RMB   80       SORTOUT_FILE path name
   D 01A5          INDISP  RMB   1        SORTIN_FILE disposition
   D 01A6          OTDISP  RMB   1        SORTOUT_FILE disposition
   D 01A7          WKDISP  RMB   1        SORT_WORK_FILE disposition
```

```
   D 01A8          JREC    RMB   KLMAX    Fully concatenated key record
   D 02A8          ZJREC   RMB   KLMAX    Fully concatenated key record
   D 03A8          MREC    RMB   RECMAX   SORTIN_FILE and SORTOUT_FILE record storage area
          * Set the stack area such that it occupies a minimum
          * of one page and extends to the next higher page boundary
   D 07A8          STKPTR  EQU   *
   D 07A8          STACK   RMB   ((STKPTR+$0200)&$FF00)-STKPTR
   D 0900          STOR    EQU   *              End of data area pointer
          *
          * Module entry point
          *
          * Initialize key save area
          *
     010B 3436     ENTRY   PSHS  D,Y,X
     010D 4F               CLRA
     010E 5F               CLRB
     010F 1F02             TFR   D,Y
     0111 30C810           LEAX  SKEYS,U
     0114 8614             LDA   #20
     0116 10AF84   ENT1    STY   0,X            clear 40 byte key
     0119 3002             LEAX  2,X            save area (SKEYS)
     011B 4A               DECA
     011C 26F8             BNE   ENT1
     011E 3536             PULS  D,Y,X
          *
          * Find out if sort control is from file or terminal
          *
     0120 10830001         CMPD  #1             if REG D
     0124 2505             BLO   TERMO          is empty then input is from terminal
     0126 1701EF           LBSR  SKIPSP         else find beginning of path description
     0129 2403             BCC   FINDEL         and then go to open this file
     012B 4F       TERMO   CLRA                 else default to terminal input
     012C 2009             BRA   TERM1
          *
          * Open CONTROL_FILE path
          *
     012E 8601     FINDEL  LDA   #1             open CONTROL_FILE
     0130 103F84           OS9   I$OPEN         for read only
     0133 10250540         LBCS  ERROR          if error in open, report it!
     0137 A7C84E   TERM1   STA   JPATH,U        else save path number
          *
          * Get SORTIN_FILE path
          *
     013A A6C84E           LDA   JPATH,U        is input of parameters
     013D 2610             BNE   SIN1           from terminal? if not skip prompt
     013F 308DFF20         LEAX  MSTR,PCR       else send
     0143 108E0015         LDY   #21            prompt
     0147 8601             LDA   #1             to
     0149 103F8A           OS9   I$WRIT         terminal
     014C A6C84E           LDA   JPATH,U
     014F 30C900B5 SIN1    LEAX  MPBUFF,U       get input file name buffer address
     0153 108E0050         LDY   #80            and then
     0157 103F8B           OS9   I$READLN       read input
     015A 1025052A         LBCS  ERROR          if bad input, report it!
     015E 30C900B5         LEAX  MPBUFF,U
     0162 170183           LBSR  SKIPSP         else find first non-space character
     0165 1025051D         LBCS  ERROR          if no input go to error exit
     0169 AFC848           STX   MASTER,U       else save address of SORTIN_FILE path name
          *
          * Get disposition for SORTIN_FILE (default is keep)
          *
     016C 17020E           LBSR  PARSE          get first character of disposition
     016F 84DF             ANDA  #%11011111     fold to upper case
     0171 8144             CMPA  #'D            if not a
     0173 2702             BEQ   SIN2           D then
     0175 864B             LDA   #'K            default the disposition to keep
     0177 A7C901A5 SIN2    STA   INDISP,U       and save it in the input disposition field
          *
          * Get SORT_WORK_FILE path
          *
     017B A6C84E           LDA   JPATH,U        is input of parameters
     017E 2610             BNE   SWORK1         from terminal? if not skip prompt
     0180 308DFF09         LEAX  SWORK,PCR      else send
     0184 108E0015         LDY   #21            prompt
     0188 8601             LDA   #1             to
     018A 103F8A           OS9   I$WRIT         terminal
     018D A6C84E           LDA   JPATH,U
     0190 30C90105 SWORK1  LEAX  ISBUFF,U       get work file name buffer address
     0194 108E0050         LDY   #80            and then
     0198 103F8B           OS9   I$READLN       read input
     019B 102504E5         LBCS  ERROR          if bad input, report it!
     019F 30C90105         LEAX  ISBUFF,U
     01A3 170177           LBSR  SKIPSP         else find first non-space character
     01A6 102504D8         LBCS  ERROR          if no input go to error exit
     01AA AFC846           STX   INDEX,U        else save address of SORT_WORK_FILE path name
          *
          * Get SORT_WORK_FILE disposition (default is Delete)
          *
     01AD 1701CD           LBSR  PARSE          get first character of disposition
     01B0 84DF             ANDA  #%11011111     fold to upper case
     01B2 814B             CMPA  #'K            if not a
     01B4 2702             BEQ   SWORK2         K then
     01B6 8644             LDA   #'D            default the disposition to Delete
     01B8 A7C901A7 SWORK2  STA   WKDISP,U       and save it in the work file disposition field
          *
```

```
                       * Get SORTOUT_FILE path
                       *
01BC A6C84E                    LDA    IPATH,U      is input of parameters
01BF 2610                      BNE    SOUT1        from terminal? if not skip prompt
01C1 308DFE83                  LEAX   OUTP,PCR     else send
01C5 108E0015                  LDY    #21          prompt
01C9 8601                      LDA    #1           to
01CB 103F8C                    OS9    I$WRITLN     terminal
01CE A6C84E                    LDA    IPATH,U
01D1 30C90155         SOUT1    LEAX   OTBUFF,U     get output file name buffer address
01D5 108E0050                  LDY    #80          and then
01D9 103F8B                    OS9    I$READLN     read input
01DC 102504A4                  LBCS   ERROR        if bad input, report it!
01E0 30C90155                  LEAX   OTBUFF,U
01E4 170131                    LBSR   SKIPSP       else find first non-space character
01E7 10250499                  LBCS   ERROR        if nothing but spaces go to error exit
01EB AFC84A                    STX    OUTPUT,U     else save address of SORTOUT_FILE path name
                       *
                       * Get SORTOUT_FILE disposition (default is keep)
                       *
01EE 17018C                    LBSR   PARSE        get first character of disposition
01F1 84DF                      ANDA   #%11011111   fold to upper case
01F3 8144                      CMPA   #'D          if not a
01F5 2702                      BEQ    SOUT2        D then
01F7 864B                      LDA    #'K          default the disposition to Keep
01F9 A7C901A6         SOUT2    STA    OTDISP,U     and save in output file disposition field
                       *
                       * Get record length
                       *
01FD A6C84E                    LDA    IPATH,U      is input of parameters
0200 2610                      BNE    RL1          from terminal? if not skip prompt
0202 308DFE9C                  LEAX   RL,PCR       else send
0206 108E0015                  LDY    #21          prompt
020A 8601                      LDA    #1           to
020C 103F8C                    OS9    I$WRITLN     terminal
020F A6C84E                    LDA    IPATH,U
0212 30C845           RL1      LEAX   BUFF,U       get general buffer address
0215 108E0050                  LDY    #80          and then
0219 103F8B                    OS9    I$READLN     read input
021C 10250464                  LBCS   ERROR        if bad input, report it!
0220 30C845                    LEAX   BUFF,U       else
0223 1700F2                    LBSR   SKIPSP       find first non-space character
0226 1025045A                  LBCS   ERROR        if no input go to error exit
022A 170102                    LBSR   CNVRT        convert ASCII number to binary
022D 10250453                  LBCS   ERROR        if bad conversion go to error exit
0231 EC45                      LDD    RESULT,U     else save length in
0233 10830400                  CMPD   #MRECMAX     is record length less than max allowed
0237 2305                      BLS    RL2          if yes then continue
0239 C64D                      LDB    #77          set error code
023B 160446                    LBRA   ERROR        and go report it!
023E EDC838           RL2      STD    LRECL,U      binary in record length save area
                       *
                       * Get sort keys (position,length,sequence)
                       *
0241 A6C84E                    LDA    IPATH,U      is input of parameters
0244 2610                      BNE    SK1          from terminal? if not skip prompt
0246 308DFE6D                  LEAX   SK,PCR       else send
024A 108E0029                  LDY    #41          prompt
024E 8601                      LDA    #1           to
0250 103F8C                    OS9    I$WRITLN     terminal
0253 170680                    LBSR   NEWLN
0256 6FC83A           SK1      CLR    TRECL,U      clear fully concatenated
0259 6FC83B                    CLR    TRECL+1,U    key length counter
025C 30C810                    LEAX   SKEYS,U      get key storage address
025F AFC83C           GETKEY   STX    LKEYP,U      set pointer to key position save area
0262 3002                      LEAX   2,X
0264 AFC83E                    STX    LKEY,U       set pointer to key length save area
0267 3001                      LEAX   1,X
0269 AFC840                    STX    SORTSQ,U     set pointer to sequence save area
026C 3001                      LEAX   1,X
026E AFC842                    STX    TYPE,U       set pointer to type save area
0271 3001                      LEAX   1,X
0273 AFC844                    STX    KEYS,U
0276 A6C84E                    LDA    IPATH,U      get input
0279 30C845                    LEAX   BUFF,U       line and
027C 108E0050                  LDY    #80          store in
0280 103F8B                    OS9    I$READLN     general buffer
0283 102503FD                  LBCS   ERROR        if input error, report it!
                       *
                       * Get key position, convert to binary, and store in SKEYS
                       *
0287 30C845                    LEAX   BUFF,U       get beginning of buffer address
028A 17008B                    LBSR   SKIPSP       find first non-space character
028D 102503F3                  LBCS   ERROR        if no input, go report it!
0291 170112                    LBSR   ENDCK        go check for input delimiter
0294 256B                      BCS    ENTEST       if present go clean up input & go build index
0296 170096                    LBSR   CNVRT1       else convert key position string to binary
0299 102503E7                  LBCS   ERROR        if bad conversion, report it!
029D 10AE45                    LDY    RESULT,U     else save
02A0 10A3C838                  CMPY   LRECL,U
02A4 2305                      BLS    GE11
02A6 C64D                      LDB    #77
02A8 1603D9                    LBRA   ERROR
02AB 313F             GE11     LEAY   -1,Y         key position
02AD 10AFD83C                  STY    [LKEYP,U]    in SKEYS via pointer

                       *
                       * Get key length, convert to binary, and store in SKEYS
                       *
02B1 1700C9                    LBSR   PARSE        get first position of next field
02B4 810D                      CMPA   #$0D         if no more input go
02B6 2605                      BNE    KL1          report it!
02B8 C610                      LDB    #16
02BA 1603C7                    LBRA   ERROR        else
02BD 8D70             KL1      BSR    CNVRT1       convert key length string to binary
02BF 102503C1                  LBCS   ERROR        if bad conversion, report it!
02C3 EC45                      LDD    RESULT,U     else save key
02C5 E7D83E                    STB    [LKEY,U]     position in SKEYS via pointer
02C8 E3D83C                    ADDD   [LKEYP,U]
02CB 10A3C838                  CMPD   LRECL,U
02CF 2305                      BLS    KL2
02D1 C64D                      LDB    #77
02D3 1603AE                    LBRA   ERROR
                       *
                       * Accumulate total key length
                       *
02D6 A6C83A           KL2      LDA    TRECL,U      get total key length counter
02D9 ABD83E                    ADDA   [LKEY,U]     add key length
02DC A7C83A                    STA    TRECL,U      and save new total key length
                       *
                       * Get key sort sequence and store in SKEYS
                       *
02DF 17009B                    LBSR   PARSE        get first position of next field
02E2 84DF                      ANDA   #%11011111   fold to upper case
02E4 8144                      CMPA   #'D
02E6 2702                      BEQ    SRTS1        if sequence not Descending
02E8 8641                      LDA    #'A          then default to Ascending sequence
02EA A7D840           SRTS1    STA    [SORTSQ,U]   and save in SKEYS via pointer
                       *
                       * Get sort type and store in SKEYS
                       *
02ED 17008D                    LBSR   PARSE        get first position of next field
02F0 84DF                      ANDA   #%11011111   fold to upper case
02F2 8142                      CMPA   #'B          if not
02F4 2702                      BEQ    TYPE1        Binary then
02F6 8643                      LDA    #'C          default the type to Character sort
02F8 A7D842           TYPE1    STA    [TYPE,U]     and save in SKEYS via pointer
02FB AEC844                    LDX    KEYS,U
02FE 16FF5E                    LBRA   GETKEY       go get next set of key data
                       *
                       * Add two bytes to total key length for record number
                       * and 1 byte for carriage return character
                       *
0301 E6C83A           ENTEST   LDB    TRECL,U      add three
0304 5C                        INCB                bytes to
0305 5C                        INCB                fully concatenated
0306 5C                        INCB                key length for record pointer
0307 E7C83A                    STB    TRECL,U      and end of record indicator
030A 4F                        CLRA
030B 10830100                  CMPD   #CLMAX       is total key length too long?
030F 102300E1                  LBLS   BUILD        if not go to build routine, else
0313 C64D                      LDB    #77          load error code and
0315 16036C                    LBRA   ERROR        report it!
                       *
                       * Find first non space character in string pointed to by reg X
                       * set carry if carrage return character is reached before any
                       * other non-space characters are encountered. else clear carry
                       * and leave REG X pointing to character found.
                       *
0318 C650             SKIPSP   LDB    #80          set max line length
031A A680             FLOOP    LDA    ,X+          get character from line
031C 810D                      CMPA   #$0D         if it is a carrage return
031E 2707                      BEQ    FS2          go set carry
0320 8120                      CMPA   #$20         else if it is a space
0322 2606                      BNE    FS1          go clear carry
0324 5A                        DECB
0325 26F3                      BNE    FLOOP        if less than max line length, go do again
0327 1A01             FS2      ORCC   #%00000001   else set carry
0329 39                        RTS                 return
032A 301F             FS1      LEAX   -1,X         move pointer back to character found
032C 1CFE                      ANDCC  #%11111110   and clear carry
032E 39                        RTS                 return
                       *
                       * Convert up to 5 ASCII bytes to 16 bit binary number.
                       * Conversion is terminated by a space, comma, or end of line.
                       * Beginning of ASCII string is pointed to by REG X on entry.
                       * Non-numeric characters and fields containing more than 5 characters
                       * will cause an error and set the carry indicator.
                       * Reg X is left pointing to beginning of next field
                       * and RESULT contains binary value.
                       *
032F 6F45             CNVRT1   CLR    RESULT,U     clear receiving
0331 6F46                      CLR    RESULT+1,U   field
0333 318D05F0                  LEAY   CONTAB,PCR   and get conversion factors
0337 5F                        CLRB
                       *
                       * Find least significant digit and verify that all characters
                       * in field are valid numeric ASCII characters
                       *
0338 A680             CNVRT11  LDA    ,X+          get a character from string
033A 812C                      CMPA   #',          if it is a comma
033C 2717                      BEQ    CNVRT12      go convert field
```

```
033E 8120            CMPA  #$20       else if it is a space
0340 2713            BEQ   CNVRT2     go convert field
0342 810D            CMPA  #$0D       else if it is a carrage return
0344 270F            BEQ   CNVRT2     go convert field
0346 8130            CMPA  #$30       else if it is less than zero
0348 252C            BLO   CNVRT4     go set carry and return
034A 8139            CMPA  #$39       else if it is greater than nine
034C 2228            BHI   CNVRT4     go set carry and return
034E 5C              INCB             else check if more than
034F C105            CMPB  #5         five characters in field
0351 23E3            BLS   CNVRT1     if less than or equal to five go do it again
0353 2021            BRA   CNVRT4     else set carry and return
          *
          * Save digit count and pointer to next field
          *
0355 E747   CNVRT2   STB   COUNT,U    get digit count
0357 301F            LEAX  -1,X       move pointer back to seperator character
0359 3410            PSHS  X          save pointer to next field
          *
          * Convert ASCII to binary
          *
035B 301F   CNVRT3   LEAX  -1,X       move pointer to lowest order digit remaining
035D 313E            LEAY  -2,Y       set REG Y to point to current digit multiplier
035F E684            LDB   0,X        get digit and
0361 C40F            ANDB  #$0F       remove left nibble
0363 4F              CLRA             and
0364 8D2B            BSR   MPY        multiply by digit multiplier
0366 ECC85C          LDD   POS+2,U    and then add
0369 E345            ADDD  RESULT,U   digit product to cumulative
036B ED45            STD   RESULT,U   total then
036D 6A47            DEC   COUNT,U    check digit count and if
036F 26EA            BNE   CNVRT3     less than 00 do it again
0371 1CFE            ANDCC %11111110  else clear carry
0373 3510            PULS  X          set REG X back to point to next field
0375 39              RTS              return
          *
          * Set error number to 16 (illegal numeric) and set carry to
          * indicate error occured.
          *
0376 C610   CNVRT4   LDB   #16        set error code
0378 1A01            ORCC  %00000001  set carry
037A 3510            PULS  X          restore stack
037C 39              RTS              return
          *
          * Scan the line for a comma and then skip all spaces
          * to next character. This character is returned in REG A.
          * If a carrage return character is encountered it is returned
          * in REG A. The selected character is pointed to by REG X on return.
          *
037D A680   PARSE    LDA   ,X+        get character
037F 812C            CMPA  #',        if it is a comma
0381 2707            BEQ   PARSE1     then go find next character
0383 810D            CMPA  #$0D       if it is not a carrage return
0385 26F6            BNE   PARSE      then go get next character
0387 301F            LEAX  -1,X       restore pointer
0389 39              RTS              else return
038A 8D0C   PARSE1   BSR   SKIPSP     skip all spaces to next character
038C 2502            BCS   PARSE2     if carrage return encountered return
038E A684            LDA   0,X        else get next character
0390 39     PARSE2   RTS              and return
          *
          * 16 BIT by 16 BIT MULTIPLY
          * Reg D contains one 16 bit number and the other 16 bit number
          * is pointed to by Reg Y.
          * The result is stored in four bytes beginning with POS.
          *
0391 ED43   MPY      STD   R1,U       save multiplier
0393 A621            LDA   1,Y        get right nibble of multiplicand and
0395 3D              MUL              multiply by right nibble of multiplier
0396 EDC85C          STD   POS+2,U    save partial product
0399 A6A4            LDA   0,Y        get left nibble of multiplicand and
039B E644            LDB   R1+1,U
039D 3D              MUL              multiply by right nibble of multiplier
039E E3C85C          ADDD  POS+2,U    add these two partial products
03A1 8900            ADCA  #0         and
03A3 EDC85B          STD   POS+1,U    save
03A6 E643            LDB   R1,U       get left nibble of multiplier
03A8 A621            LDA   1,Y        get right nibble of multiplicand
03AA 3D              MUL              multiply and
03AB EDC85F          STD   POS+5,U    save partial product
03AE A6A4            LDA   0,Y        get left nibble of multiplicand
03B0 E643            LDB   R1,U       get left nibble of multiplier
03B2 3D              MUL              multiply and
03B3 E3C85F          ADDD  POS+5,U    add these two partial
03B6 8900            ADCA  #0         products and
03B8 EDC85E          STD   POS+4,U    save
03BB A6C85C          LDA   POS+2,U    add the
03BE ABC860          ADDA  POS+6,U    two previously saved
03C1 A7C85C          STA   POS+2,U    partial products
03C4 A6C85B          LDA   POS+1,U    to arrive
03C7 A9C85F          ADCA  POS+5,U    at the
03CA A7C85B          STA   POS+1,U    complete
03CD A6C85E          LDA   POS+4,U    thirty two
03D0 8900            ADCA  #0         bit product and
03D2 A7C85A          STA   POS,U      save it at POS
03D5 39              RTS              then return
          *
          * Check for input delimiter (END) and set carry if found.
          * It can be either upper or lower case characters.
          *
03D6 A684   ENDCK    LDA   0,X        get first character
03D8 84DF            ANDA  %11011111  fold to upper case
03DA 8145            CMPA  #'E        is it an E
03DC 2613            BNE   NOTEND     if not clear carry and return
03DE A601            LDA   1,X        else get next character
03E0 84DF            ANDA  %11011111  fold to upper case
03E2 814E            CMPA  #'N        is it a N
03E4 260B            BNE   NOTEND     if not clear carry and return
03E6 A602            LDA   2,X        else get next character
03E8 84DF            ANDA  %11011111  fold to upper case
03EA 8144            CMPA  #'D        is it a D
03EC 2603            BNE   NOTEND     if not clear carry and return
03EE 1A01            ORCC  %00000001  else set carry and
03F0 39              RTS              return
03F1 1CFE   NOTEND   ANDCC %11111110  clear carry and
03F3 39              RTS              return
          *
          * Build SORT_WORK_FILE routine
          *
          * Create SORT_WORK_FILE
          *
03F4 30D846 BUILD    LEAX  [TMDEL,U]  get work file path name
03F7 C603            LDB   #%00000011 and set up to
03F9 8602            LDA   #2         create this
03FB 103F83          OS9   I$CREA     file
03FE 10250282        LBCS  ERROR      if problem, report it!
0402 A7C84F          STA   OPATH,U    else save path number
          *
          * Open SORTIN_FILE for read only
          *
0405 30D84B          LEAX  [MASTER,U] get sort input path name
0408 8601            LDA   #1         and open it
040A 103F84          OS9   I$OPEN     for read only
040D 10250273        LBCS  ERROR      if problem, report it!
0411 A7C850          STA   MPATH,U    else save path number and
0414 860D            LDA   #$0D       load end of
0416 A742            STA   CR,U       record value and then
0418 6FC4            CLR   R1,U       clear
041A 6F41            CLR   R1+1,U     record counter
          *
          * Get first set of key parameters
          *
041C 30C810 BLOOP    LEAX  SKEYS,U
041F AFC83C          STX   LKEYP,U    save address of key position
0422 3002            LEAX  2,X
0424 AFC83E          STX   LKEY,U     save address of key length
0427 3003            LEAX  3,X
0429 AFC844          STX   KEYS,U
042C A6C850          LDA   MPATH,U    get SORTIN_FILE path number
042F 30C903A0        LEAX  MREC,U     and then
0433 10AEC83A        LDY   LRECL,U    read a record
0437 103F89          OS9   I$READ     storing it in MREC
043A 254E            BCS   CKEOF      if carry set go check if EOF condition
043C AEC4            LDX   R1,U       else
043E 3001            LEAX  1,X        increment record
0440 AFC4            STX   R1,U       counter
          *
          * Get data pointed to by key and save in SORT_WORK_FILE
          *
0442 30C903A0 BLOOP0 LEAX  MREC,U     get beginning address of sort input record
0446 1F10            TFR   X,D        and
0448 E3D83C          ADDD  [LKEYP,U]  add key position to it
044B 1F01            TFR   D,X        then write
044D E6D83E          LDB   [LKEY,U]   the
0450 4F              CLRA             number
0451 1F02            TFR   D,Y        of
0453 A6C84F          LDA   OPATH,U    bytes specified by the
0456 103F8A          OS9   I$WRIT     key length to the work file
0459 10250227        LBCS  ERROR      if bad write, report it!
          *
          * Get next set of key parameters
          *
045D 3410            PSHS  X          save position pointer
045F AEC844          LDX   KEYS,U     get key data save pointer
0462 AFC83C          STX   LKEYP,U    set new key position pointer
0465 3002            LEAX  2,X
0467 AFC83E          STX   LKEY,U     set new key length pointer
046A 3003            LEAX  3,X
046C AFC844          STX   KEYS,U     save new key data save pointer
046F 3510            PULS  X          restore position pointer
0471 E6D83E          LDB   [LKEY,U]   get new key length
          *
          * If key length is zero then no more sets of sort key
          * parameters are available else go back and get key data
          *
0474 2702            BEQ   BLOOP1     if zero all keys have been processed
0476 20CA            BRA   BLOOP0     else go get next key data
          *
          * After last key is saved then add record number and EOF
          * character to SORT_WORK_FILE record.
          *
```

```
0478 30C4      BLOOP1  LEAX  R1,U       get record number and EOF character
047A 108E0003          LDY   #3         and
047E A6C84F            LDA   IPATH,U    write
0481 103F8A            OS9   I$WRIT     to sort file
0484 102501FC          LBCS  ERROR      if bad write, report it!
0488 2092              BRA   BLOOP      else go get next record from SORTIN_FILE
     *
     * Check to see if error condition is end of file or real error
     *
048A 4F        CKEOF   CLRA
048B 10830003          CMPD  #211       if error code not equal end of file code
048F 102601F1          LBNE  ERROR      then go to error exit.
     *
     * Close SORT_WORK_FILE and SORTIN_FILE
     *
0493 ECC4              LDD   R1,U       else save record
0495 ED48              STD   SEQ,U      count and then
0497 A6C84F            LDA   IPATH,U    close the
049A 103F8F            OS9   I$CLOS     work file
049D 102501E3          LBCS  ERROR      if close error, report it!
04A1 A6C850            LDA   APATH,U    else close
04A4 103F8F            OS9   I$CLOS     the sort input file
04A7 102501D9          LBCS  ERROR      if close error, report it!

     *
     * Main sort routine
     *
     * Open SORT_WORK_FILE for update
     *
04AB 30D846    SORT    LEAX  [INDEX1,U] get SORTWORK path description
04AE 8603              LDA   #3         and
04B0 103F84            OS9   I$OPEN     open it for update
04B3 102501CD          LBCS  ERROR      if an error on open, report it
04B7 A7C84F            STA   IPATH,U    else save path number
     *
     * G1=INT(R1/2)
     *
04BA ECC4              LDD   R1,U       get number of records
04BC 44                LSRA             and divide
04BD 56                RORB             it by two
04BE 1CFE              ANDCC #%11111110 and clear carry
04C0 EDC852            STD   G1,U       then place result in G1
     *
     * I=G1+1
     *
04C3 AEC852    ILOOP   LDX   G1,U       get G1
04C6 3001              LEAX  1,X        and increment it by one
04C8 AFC854            STX   I,U        then save in I
     *
     * J=I-G1
     *
04CB ECC854    JLOOP   LDD   I,U        get I
04CE A3C852            SUBD  G1,U       and subtract G1 from it
04D1 EDC856            STD   J,U        then save in J
     *
     * J1=J+G1
     *
04D4 ECC856    J1LOOP  LDD   J,U        get J
04D7 E3C852            ADDD  G1,U       and add G1 to it
04DA EDC858            STD   J1,U       then save in J1
     *
     * GET RECORD J and store in JREC
     *
04DD AEC856            LDX   J,U        get pointer to J record
04E0 AFC861            STX   REC,U      and place in REC
04E3 30C90140          LEAX  JREC,U     put buffer address to load in
04E7 AFC863            STX   RECP,U     buffer pointer area RECP
04EA 8D7D              BSR   GETREC     go get record J and place in JREC buffer
04EC 10250194          LBCS  ERROR      if an error, go report it
     *
     * GET RECORD J1 and store in J1REC
     *
04F0 AEC858            LDX   J1,U       get pointer to J1 record
04F3 AFC861            STX   REC,U      and place in REC
04F6 30C90240          LEAX  J1REC,U    put buffer address to load in
04FA AFC863            STX   RECP,U     buffer pointer area RECP
04FD 8D6A              BSR   GETREC     go get record J1 and place in J1REC buffer
04FF 10250181          LBCS  ERROR      if an error, go report it
     *
     * COMPARE J1REC TO JREC
     *
0503 1700C0            LBSR  COMPAR     go compare buffers JREC with J1REC
0506 2531              BCS   INCI       if carry set then skip record swap
     *
     * SWAP JREC WITH J1REC
     *
0508 30C90140  SWAP    LEAX  JREC,U     get address of JREC
050C AFC863            STX   RECP,U     and place in RECP pointer
050F AEC858            LDX   J1,U       get J1 record pointer
0512 AFC861            STX   REC,U      and place in REC pointer
0515 8D77              BSR   PUTREC     go put JREC buffer in J1REC position in file
0517 10250169          LBCS  ERROR      if error, report it
051B 30C90240          LEAX  J1REC,U    get address of J1REC
051F AFC863            STX   RECP,U     and place in RECP pointer
0522 AEC856            LDX   J,U        get J record pointer
0525 AFC861            STX   REC,U      and place in REC pointer
0528 8D64              BSR   PUTREC     go put J1REC in JREC position in file
052A 10250156          LBCS  ERROR      if error, report it
     *
     * J=J-G1
     *
052E ECC856            LDD   J,U        get J pointer
0531 A3C852            SUBD  G1,U       and subtract G1 from it
0534 EDC856            STD   J,U        and save in J
     *
     * IS J>0
     *
0537 249B              BHS   J1LOOP     if J is greater than or equal to zero go reset J1
0539 AEC854    INCI    LDX   I,U        else get I
053C 3001              LEAX  1,X        and increment it by one
053E AFC854            STX   I,U        then save it in I
     *
     * IS I<=R1
     *
0541 ECC854            LDD   I,U        get I
0544 A3C4              SUBD  R1,U       and subtract R1 from it
0546 2383              BLS   JLOOP      if result is zero or negative go reset J & J1
     *
     * G1 = INT(G1/2)
     *
0548 ECC852            LDD   G1,U       get G1
054B 44                LSRA             divide it
054C 56                RORB             by two
054D 1CFE              ANDCC #%11111110 and clear carry
054F EDC852            STD   G1,U       then save result in G1
     *
     * IS G1>=1
     *
0552 ECC852            LDD   G1,U       get G1
0555 830001            SUBD  #1         and if it is less than
0558 1024FF67          LBHS  ILOOP      or equal to one go reset I & J & J1
     *
     * RETURN
     *
055C A6C84F            LDA   IPATH,U    else get SORTWORK file path number
055F 103F8F            OS9   I$CLOS     and close the file
0562 1025011E          LBCS  ERROR      if an error, go report it
0566 16011E            LBRA  REBLD      go to generate SORTOUT file
     *
     * Get record from SORT_WORK_FILE
     *
0569 8D48      GETREC  BSR   BYTEP      locate position of selected record
056B A6C84F            LDA   IPATH,U    and
056E AEC85A            LDX   POS,U      set up
0571 10AEC85C          LDY   POS+2,U    to have OS9
0575 1E23              EXG   Y,U        position the file
0577 103F88            OS9   I$SEEK     pointer to this record
057A 2511              BCS   GETRET     if a seek error, report it!
057C 1E23              EXG   Y,U        else
057E E6C83A            LDB   IRECL,U    get set
0581 4F                CLRA             to
0582 1F02              TFR   D,Y        read
0584 A6C84F            LDA   IPATH,U    the record
0587 30D863            LEAX  [RECP,U]   from the file and
058A 103F89            OS9   I$READ     place it in the field pointed to by RECP
058D 39        GETRET  RTS              and return
     *
     * Put a record back into the SORT_WORK_FILE
     *
058E 8D23      PUTREC  BSR   BYTEP      locate position of selected record
0590 A6C84F            LDA   IPATH,U    and
0593 AEC85A            LDX   POS,U      set up
0596 10AEC85C          LDY   POS+2,U    to have OS9
059A 1E23              EXG   Y,U        position the file
059C 103F88            OS9   I$SEEK     pointer to this record
059F 2511              BCS   PUTRET     if seek error, report it!
05A1 1E23              EXG   Y,U        else
05A3 E6C83A            LDB   IRECL,U    get set
05A6 4F                CLRA             to
05A7 1F02              TFR   D,Y        write
05A9 A6C84F            LDA   IPATH,U    the record
05AC 30D863            LEAX  [RECP,U]   from the field pointed to by RECP
05AF 103F8A            OS9   I$WRIT     to the sort work file
05B2 39        PUTRET  RTS
     *
     * Multiply record length(IRECL) by record number(REC)
     *
05B3 10AEC861  BYTEP   LDY   REC,U      get record number
05B7 2710              BEQ   BYTRET     if zero then return
05B9 313F              LEAY  -1,Y       else change to base zero
05BB 10AFC861          STY   REC,U
05BF E6C83A            LDB   IRECL,U    get record length
05C2 4F                CLRA             and go
05C3 31C861            LEAY  REC,U
05C6 17F0C0            LBSR  MP1        multiply and
05C9 39        BYTRET  RTS              then return
     *
     * COMPARE J1REC AND JREC
     * Set carry if JREC <= J1REC for ascending sorts
     * Set carry if JREC >= J1REC for descending sorts
```

```
                   *
05CA 3436          COMPAR  PSHS  D,X,Y
05CC 30C810                LEAX  SKEYS,U     get key data stack
05CF A604                  LDA   4,X         get key sort type
05D1 A7C84C                STA   TSORT,U     and save
05D4 E602                  LDB   2,X         get key length and save in REG B
05D6 A603                  LDA   3,X         then save the sort sequence in REG A
05D8 AFC844                STX   KEYS,U      save key data pointer
05DB 30C901A0              LEAX  JREC,U      set REG X to point to beginning of JREC
05DF 31C902A0              LEAY  JIREC,U     set REG Y to point to beginning of JIREC
                   *
                   * Is the sort sequence DESCENDING ?
                   *
05E3 8144          COM1    CMPA  #'D         is sort sequence descending?
05E5 2747                  BEQ   COMPL2      if so go do descending compare, else default to a
                   *
                   * Ascending sort comparison
                   *
05E7 A6C84C        COMPL1  LDA   TSORT,U     get key sort type
05EA 8142                  CMPA  #'B         if it is binary
05EC 2726                  BEQ   COMPT1      continue
05EE A6A4                  LDA   0,Y         else get character
05F0 48                    ASLA              and shift one bit to left
05F1 8580                  BITA  #%10000000 is alpha bit set
05F3 2708                  BEQ   COM10       if no go save it
05F5 8540                  BITA  #%01000000 else check lower case bit
05F7 2704                  BEQ   COM10       if it is not set go save it
05F9 8A01                  ORA   #%00000001 else set low order bit
05FB 84DF                  ANDA  #%11011111 and clear lower case bit
05FD A7C84D        COM10   STA   COMPSV,U    and save it
0600 A684                  LDA   0,X         then get the second character
0602 48                    ASLA              and shift one bit to left
0603 8580                  BITA  #%10000000 is alpha bit set
0605 2708                  BEQ   COM20       if no go to compare
0607 8540                  BITA  #%01000000 is lower case bit set
0609 2704                  BEQ   COM20       if no go to compare
060B 8A01                  ORA   #%00000001 else set low order bit
060D 84DF                  ANDA  #%11011111 and clear lower case bit
060F A1C84D        COM20   CMPA  COMPSV,U    compare a character from JREC to one from JIREC
0612 2004                  BRA   COMPT2       go test result
0614 A684          COMPT1  LDA   0,X         get a character from JREC
0616 A1A4                  CMPA  0,Y         then compare it to a character from JIREC
0618 2290          COMPT2  BHI   CLRCAR      if JREC > JIREC go clear carry and return
061A 2563                  BLO   SETCAR      else if JREC < JIREC go set carry and return
061C 3001                  LEAX  1,X         else get next
061E 3121                  LEAY  1,Y         set of characters
0620 5A                    DECB              and decrement character counter
0621 26C4                  BNE   COMPL1      if counter not zero go do another compare
0623 2040                  BRA   COM2        else get next set of keys
0625 1CFE          CLRCAR  ANDCC #%11111110 clear carry
0627 2058                  BRA   COMRET      and return
                   *
                   * Descending sort comparison
                   *
0629 A6C84C        COMPL2  LDA   TSORT,U     get requested sort type
062C 8142                  CMPA  #'B         is it binary
062E 2726                  BEQ   COMPT3      if so skip upper-lower case fold
0630 A6A4                  LDA   0,Y         else get character from JIREC
0632 48                    ASLA              and shift one bit to left
0633 8580                  BITA  #%10000000 is alpha bit set
0635 2708                  BEQ   COM30       if not go save it
0637 8540                  BITA  #%01000000 else check lower case bit
0639 2704                  BEQ   COM30       if it is not set go save it
063B 8A01                  ORA   #%00000001 else set low order bit
063D 84DF                  ANDA  #%11011111 and clear lower case bit
063F A7C84D        COM30   STA   COMPSV,U    save in temp area
0642 A684                  LDA   0,X         get character from JREC
0644 48                    ASLA              and shift one bit to left
0645 8580                  BITA  #%10000000 is alpha bit set
0647 2708                  BEQ   COM40       if not go compare
0649 8540                  BITA  #%01000000 is lower case bit set
064B 2704                  BEQ   COM40       if no go to compare
064D 8A01                  ORA   #%00000001 else set low order bit
064F 84DF                  ANDA  #%11011111 and clear lower case bit
0651 A1C84D        COM40   CMPA  COMPSV,U    compare a character from JREC to one from JIREC
0654 2004                  BRA   COMPT4      go test result
0656 A684          COMPT3  LDA   0,X         get a character from JREC
0658 A1A4                  CMPA  0,Y         compare it against a character from JIREC
065A 2223          COMPT4  BHI   SETCAR      if JREC > JIREC go set carry and return
065C 25C7                  BLO   CLRCAR      else if JREC < JIREC go clear carry and return
065E 3001                  LEAX  1,X         else get next set of
0660 3121                  LEAY  1,Y         characters and
0662 5A                    DECB              decrement character counter
0663 26C4                  BNE   COMPL2      if counter not zero then go do another compare
                   *
                   * Increment to next key
                   *
0665 3410          COM2    PSHS  X           else
0667 AEC844                LDX   KEYS,U      get key data pointer
066A 3005                  LEAX  5,X         and increment to next
066C AFC844                STX   KEYS,U      key data set
066F A604                  LDA   4,X         get sort type
0671 A7C84C                STA   TSORT,U     and save
0674 A603                  LDA   3,X         save sort sequence in REG A
0676 E602                  LDB   2,X         save key length in REG B
```

```
0678 3510                  PULS  X
067A 5D                    TSTB
067B 1026FF64              LBNE  COM1        if key length is not zero go do another compare
067F 1A01          SETCAR  ORCC  #%00000001 else set carry for equal condition
0681 3536          COMRET  PULS  A,B,X,Y     and
0683 39                    RTS               return
                   *
                   * Error exit routine
                   * All errors are standard OS9 error codes
                   *
0684 103F06        ERROR   OS9   F$EXIT      Return to OS-9 with error code
                   *
                   * Check to see if SORTOUT_FILE is to be kept
                   * if not then do not create it
                   *
0687 A6C901A6      RESLB   LDA   OTDISP,U    get SORTOUT_FILE disposition
068B 8144                  CMPA  #'D         and if Delete
068D 102700C1              LBEQ  RESLB1      do not create it
                   *
                   * Build SORTOUT_FILE in sorted order
                   *
0691 6F48                  CLR   SEQ,U       else clear
0693 6F49                  CLR   SEQ+1,U     record counter
                   *
                   * Open SORT_WORK_FILE for read
                   *
0695 30C8A0                LEAX  INDEX,U     get SORT_WRK_FILE path name pointer
0698 8601                  LDA   #1          and
069A 103F84                OS9   I$OPEN      open for read only
069D 25E5                  BCS   ERROR       if open error, report it!
069F A7C84F                STA   IPATH,U     else save path number
                   *
                   * Open SORTIN_FILE for read
                   *
06A2 30C840                LEAX  MASTER,U    get SORTIN_FILE path name pointer
06A5 8601                  LDA   #1          and
06A7 103F84                OS9   I$OPEN      open for read only
06AA 25D8                  BCS   ERROR       if open error, report it!
06AC A7C850                STA   MPATH,U     else save path number
                   *
                   * Create SORTOUT_FILE
                   *
06AF 30C8A0                LEAX  OUTPUT,U    get SORTOUT_FILE path name pointer
06B2 C603                  LDB   #%00000011 and
06B4 8602                  LDA   #2          create
06B6 103F83                OS9   I$CREA      for write only
06B9 25C9                  BCS   ERROR       if create error, report it!
06BB A7C851                STA   SPATH,U     else save path number
                   *
                   * Get a record from SORT_WORK_FILE
                   *
06BE 30C901A0      RLOOP   LEAX  JREC,U      get SORT_WORK_FILE buffer pointer
06C2 E6C83A                LDB   IRECL,U     set SORT_WORK_FILE record length
06C5 4F                    CLRA
06C6 1F02                  TFR   D,Y         and
06C8 A6C84F                LDA   IPATH,U     get
06CB 103F89                OS9   I$READ      a record from SORT_WORK_FILE
06CE 2550                  BCS   RLEOF       if carry set go see if end of file
06D0 AE48                  LDX   SEQ,U       else
06D2 3001                  LEAX  1,X         increment record
06D4 AF48                  STX   SEQ,U       counter and save it
                   *
                   * Get SORTIN record number from SORT_WORK_FILE
                   *
06D6 31C901A0              LEAY  JREC,U      point to beginning of SORT_WORK_FILE record
06DA E6C83A                LDB   IRECL,U     get SORT_WORK_FILE record length
06DD 5A                    DECB              move
06DE 5A                    DECB              pointer
06DF 5A                    DECB              to SORTIN_FILE record pointer
06E0 31A5                  LEAY  B,Y         get
06E2 ECA4                  LDD   0,Y         SORTIN_FILE
06E4 1F01                  TFR   D,X         record pointer
06E6 301F                  LEAX  -1,X        and convert to base zero
06E8 1F10                  TFR   X,D         and
06EA EDA4                  STD   0,Y         set REG Y to point to the record pointer
06EC ECC830                LDD   LRECL,U     get SORTIN_FILE record length
06EF 17FC9F                LBSR  MPY         and multiply record length by record number
06F2 A6C850                LDA   MPATH,U     so that a
06F5 AEC85A                LDX   POS,U       seek can
06F8 10AEC85C              LDY   POS+2,U     be performed to
06FC 1E23                  EXG   Y,U         position file
06FE 103F88                OS9   I$SEEK      to beginning of desired record in SORTIN_FILE
0701 2581                  BCS   ERROR       if bad seek, report it!
0703 1E23                  EXG   Y,U         else
0705 A6C850                LDA   MPATH,U     get set to
0708 10AEC830              LDY   LRECL,U     read the
070C 30C903A0              LEAX  MREC,U      selected record
0710 103F89                OS9   I$READ      from SORTIN_FILE
0713 1025FF6D              LBCS  ERROR       if bad read, report it!
0717 30C903A0              LEAX  MREC,U      else
071B A6C851                LDA   SPATH,U     go write this
071E 10AEC830              LDY   LRECL,U     record
0722 103F8A                OS9   I$WRIT      to the SORTOUT_FILE
0725 1025FF5B              LBCS  ERROR       if bad write, report it!
0729 2093                  BRA   RLOOP       else go process next SORT_WORK_FILE record
```

```
                         *
                         * Check to see if end of file instead of real error
                         *
0728 4F          RLEOF   CLRA
072C 108300D3            CMPD  #211         if return code is not for EOF
0730 1026FF50            LBNE  ERROR        go report it!
                         *
                         * Close SORT_WORK_FILE
                         *
0734 A6C84F              LDA   IPATH,U      else
0737 103F8F              OS9   I$CLOS       close the SORT_WORK_FILE
073A 1025FF46            LBCS  ERROR        if bad close, report it!
                         *
                         * Close SORTIN_FILE
                         *
073E A6C850              LDA   RPATH,U      else
0741 103F8F              OS9   I$CLOS       close SORTIN_FILE
0744 1025FF3C            LBCS  ERROR        if bad close, report it!
                         *
                         * Close SORTOUT_FILE
                         *
0748 A6C851              LDA   SPATH,U      else close
074B 103F8F              OS9   I$CLOS       the SORTOUT_FILE
074E 1025FF32            LBCS  ERROR        if bad close, report it!
                         *
                         * Close CONTROL_FILE
                         *
0752 A6C84E      RERL01  LDA   JPATH,U      else
0755 103F8F              OS9   I$CLOS       close CONTROL_FILE
0758 1025FF28            LBCS  ERROR        if bad close, report it!
                         *
                         * Delete SORT_WORK_FILE if Delete is specified
                         *
075C A6C901A7            LDA   WKDISP,U     else get SORT_WORK_FILE disposition
0760 8144                CMPA  #'D          if not delete disposition
0762 260A                BNE   RERL02       go to check disposition of SORTIN_FILE
0764 30D846              LEAX  [INDEX,U]    else go and
0767 103F87              OS9   I$DLET       delete SORT_WORK_FILE
076A 1025FF16            LBCS  ERROR        if bad delete, report it!
                         *
                         * Delete SORTIO_FILE if Delete is specified
                         *
076E A6C901A5    RERL02  LDA   INDISP,U     else get SORTIN_FILE disposition
0772 8144                CMPA  #'D          if not delete disposition
0774 2608                BNE   LIST         go to output sort statistics
0776 30D840              LEAX  [MASTER,U]   else go and
0779 103F87              OS9   I$DLET       delete SORTIN_FILE
077C 1025FF04            LBCS  ERROR        if bad delete, report it!
                         *
                         * Print sort activity for this run
                         *
0780 862C        LIST    LDA   #',          load field
0782 A74F                STA   COMMA,U      separator and
0784 170163              LBSR  NPAGE        go to top of form
                         *
                         * Print title and copyright
                         *
0787 308DF8B2            LEAX  TITLE,PCR    output statistics report
078B 108E0030            LDY   #48          header
078F 8601                LDA   #1           message
0791 103F8A              OS9   I$WRIT       and
0794 170147              LBSR  NEWL         go to new line
                         *
                         * Print number of records sorted
                         *
0797 170160              LBSR  BTOA         convert record count to ASCII string
079A 304A                LEAX  ASCII,U      output
079C 108E0005            LDY   #5           record
07A0 8601                LDA   #1           count
07A2 103F8A              OS9   I$WRIT       followed
07A5 308DF8A7            LEAX  RCM1,PCR     by
07A9 108E000F            LDY   #15          record
07AD 8601                LDA   #1           count
07AF 103F8A              OS9   I$WRIT       message
07B2 170129              LBSR  NEWLN        skip a
07B5 170126              LBSR  NEWLN        line
                         *
                         * Print SORTIN_FILE name
                         *
07B8 308DF8A7            LEAX  SRTD,PCR     output the
07BC 108E0013            LDY   #19          SORTIN_FILE
07C0 8601                LDA   #1           message
07C2 103F8A              OS9   I$WRIT       followed
07C5 30D840              LEAX  [MASTER,U]   by the
07C8 108E0050            LDY   #80          SORTIN_FILE
07CC 8601                LDA   #1           path
07CE 103F8C              OS9   I$WRLN       description and
07D1 17010A              LBSR  NEWLN        go to a new line
                         *
                         * Print SORT_WORK_FILE name
                         *
07D4 308DF8B5            LEAX  SWRK,PCR     output the
07D8 108E0015            LDY   #21          SORT_WORK_FILE
07DC 8601                LDA   #1           message
07DE 103F8A              OS9   I$WRIT       followed
07E1 30D846              LEAX  [INDEX,U]    by the
07E4 108E0050            LDY   #80          SORT_WORK_FILE
07E8 8601                LDA   #1           path
07EA 103F8C              OS9   I$WRLN       description and
07ED 1700EE              LBSR  NEWLN        go to a new line
                         *
                         * Print SORTOUT_FILE name
                         *
07F0 308DF8B4            LEAX  OUTP,PCR     output the
07F4 108E0015            LDY   #21          SORTIN_FILE
07F8 8601                LDA   #1           message
07FA 103F8A              OS9   I$WRIT       followed
07FD 30D844              LEAX  [OUTPUT,U]   by the
0800 108E0050            LDY   #80          SORTOUT_FILE
0804 8601                LDA   #1           path
0806 103F8C              OS9   I$WRLN       description and
0809 1700D2              LBSR  NEWLN        then skip
080C 1700CF              LBSR  NEWL         a line
                         *
                         * Print record length
                         *
080F 308DF8BF            LEAX  RL,PCR       output the
0813 108E0015            LDY   #21          record length
0817 8601                LDA   #1           message
0819 103F8A              OS9   I$WRIT       and then
081C ECC858              LDD   LRECL,U      convert the
081F ED48                STD   SEQ,U        record length
0821 1700B6              LBSR  BTOA         to ASCII
0824 304A                LEAX  ASCII,U      characters
0826 108E0005            LDY   #5           and
082A 8601                LDA   #1           output
082C 103F8A              OS9   I$WRIT       it
082F 1700AC              LBSR  NEWLN        then skip
0832 1700A9              LBSR  NEWLN        a line
                         *
                         * Print sort keys
                         *
0835 308DF87E            LEAX  SK,PCR       output the
0839 108E0029            LDY   #41          sort key
083D 103F8A              OS9   I$WRIT       message
0840 17009B              LBSR  NEWL         go to a new line
0843 30C810              LEAX  SKEYS,U      get pointer to key save area
0846 AFC844              STX   KEYS,U       point to first set of key data
0849 8D1D        LIST1   BSR   PRTLN        output a set of key data
084B 170090              LBSR  NEWLN        go to new line
084E AEC844              LDX   KEYS,U       set up for
0851 E602                LDB   2,X          next set of key data
0853 26F4                BNE   LIST1        if another key data set go output it
0855 308DF8AF            LEAX  ENDMSG,PCR   output the
0859 8601                LDA   #1           END delimiter
085B 108E0003            LDY   #3           message
085F 103F8A              OS9   I$WRIT       and then
0862 8D7A                BSR   NEWLN        output one line feed
0864 5F                  CLRB               then clear error code
0865 103F06              OS9   F$EXIT       and return to OS9
                         *
                         * Print a line of key data (key position,key length,key sequence)
                         *
0868 AEC844      PRTLN   LDX   KEYS,U       point to a key data set
086B EC84                LDD   0,X          get key position
086D C30001              ADDD  #1           correct to base 1
0870 ED48                STD   SEQ,U        save corrected value
0872 3002                LEAX  2,X          and get set to
0874 AFC844              STX   KEYS,U       get key length
0877 170060              LBSR  BTOA         go convert key position
087A 304A                LEAX  ASCII,U      to ASCII
087C 108E0006            LDY   #6           and
0880 8601                LDA   #1           output it with
0882 103F8A              OS9   I$WRIT       a trailing comma
0885 AEC844              LDX   KEYS,U       get key length pointer
0888 E684                LDB   0,X          and put key length in REG B
088A 4F                  CLRA
088B ED48                STD   SEQ,U        save key length
088D 3001                LEAX  1,X          and get set to
088F AFC844              STX   KEYS,U       get key sort sequence
0892 8D44                BSR   BTOA         go convertkey length
0894 304A                LEAX  ASCII,U      to ASCII
0896 108E0006            LDY   #6           and
089A 8601                LDA   #1           output it with
089C 103F8A              OS9   I$WRIT       a trailing comma
089F AEC844              LDX   KEYS,U       get key sort sequence pointer
08A2 3001                LEAX  1,X          and set up pointer
08A4 AFC844              STX   KEYS,U       for next set of key data
08A7 A61F                LDA   -1,X         get sort sequence in REG A
08A9 308DF8C3            LEAX  ASCEND,PCR   set up default sort sequence message
08AD 8144                CMPA  #'D          if sort sequence is not a D
08AF 2604                BNE   ASCND        then go output default message
08B1 308DF8C6            LEAX  DESCND,PCR   else output descending sequence message
08B5 108E000B    ASCND   LDY   #11          set sequence message length
08B9 8601                LDA   #1           and output path number
08BB 103F8A              OS9   I$WRIT       then write message
08BE AEC844              LDX   KEYS,U       get key pointer
08C1 3001                LEAX  1,X          and increment it by one
```

```
01234  08C3 AFC844           STX   KEYS,U     then save again
01235  08C6 A61F             LDA   -1,X       get sort type
01236  08C8 308DF82A         LEAX  CTYPE,PCR  and set up default message
01237  08CC 8142             CMPA  #'B        is it a binary sort?
01238  08CE 2604             BNE   CHAR       if not use default message
01239  08D0 308DF82D         LEAX  BTYPE,PCR  else set up binary type message
01240  08D4 108E0009  CHAR   LDY   #9         load message length
01241  08D8 8601             LDA   #1         load output path number
01242  08DA 103F8A           OS9   I$WRITE    and write message
01243  08DD 39               RTS              then return
01244       *
01245       * Output one new line character and one line feed character
01246       *
01247  08DE 308DF770  NEWLN  LEAX  CRLF,PCR   output carriage return
01248  08E2 108E0002         LDY   #2         and new line
01249  08E6 8601             LDA   #1         characters
01250  08E8 103F8A           OS9   I$WRITE    to the terminal and
01251  08EB 39               RTS              return
01252       *
01253       * Output a forms feed Character followed by a null character
01254       *
01255  08EC 308DF771  NPAGE  LEAX  NP,PCR     output a top of forms
01256  08F0 108E0002         LDY   #2         character plus
01257  08F4 8601             LDA   #1         one null character to the
01258  08F6 103F8A           OS9   I$WRITE    terminal and
01259  08F9 39               RTS              return
01260       *
01261       * Convert 2 bytes of binary to 5 ASCII characters
01262       * Binary input is in SEO and receiving field is ASCII
01263       *
01264  08FA 8605      BTOA   LDA   #5         set max digit limit
01265  08FC A741             STA   R1+1,U     and save
01266  08FE EC48             LDD   SEO,U      get binary value
01267  0900 314A             LEAY  ASCII,U    point to ASCII receiving field
01268  0902 308D001F         LEAX  CONSTS,PCR point to conversion constants
01269  0906 6FC4      BTOA1  CLR   R1,U       clear factor counter
01270  0908 A384      BTOA2  SUBD  0,X        substract digit conversion factor
01271  090A 2504             BCS   BTOA3      if too much go add it back
01272  090C 6CC4             INC   R1,U       else increment factor counter
01273  090E 20F8             BRA   BTOA2      and do it again
01274  0910 E384      BTOA3  ADDD  0,X        add back overage
01275  0912 3402             PSHS  A          save left nibble of result
01276  0914 A6C4             LDA   R1,U       get current factor
01277  0916 8B30             ADDA  #$30       add ASCII offset to factor
01278  0918 A7A4             STA   0,Y        and save in receiving field
01279  091A 3502             PULS  A          recover left nibble of result
01280  091C 3121             LEAY  1,Y        point to next receiving field position
01281  091E 3002             LEAX  2,X        point to next conversion factor
01282  0920 6A41             DEC   R1+1,U     check to see if all digits have been converted
01283  0922 26E2             BNE   BTOA1      if not go do it again
01284  0924 39               RTS              else return
01285       *
01286       * Constants used for ASCII-binary conversions
01287       *
01288  0925 271003E8  CONSTS FDB   10000,1000,100,10,1
01289  0929 0064000A
01290  092D 0001                              CONSTS FDB  10000,1000,100,10,1
01291  092F           COMEND EQU   *
01292  092F 8A9E53           EMOD
01293  0932           NEEDED EQU   *
01294                        END

00000 error(s)
00000 warning(s)
$0932 02354 program bytes generated
$0900 02304 data bytes allocated
$09DF 02527 bytes used for symbols

00E0 L ASCEND   000A D ASCII    0889 L ASCND    041C L BLOOP    0442 L BLOOP0   047B L BLOOP1
08FA L BTOA     0906 L BTOA1    0908 L BTOA2    0910 L BTOA3    00FF L BTYPE    0045 D BUFF
03F4 L BUILD    05D3 L BYTEP    05C0 L BYTRET   08D4 L CHAR     0480 L CKEOF    0425 L CLRCAR
032F L CNVRT    0330 L CNVRT1   0353 L CNVRT2   0358 L CNVRT3   0374 L CNVRT4   05E3 L COM1
05FD L COM10    0663 L COM2     060F L COM20    065F L COM30    0651 L COMAND   000F D COMMA
05CD L COMPAR   05E7 L COMPL1   0629 L COMPL2   0049 D COMPDV   0614 L COMPT1   0618 L COMP12
0656 L COMPT3   065A L COMPT4   0601 L COMRET   092F E COMEND   0925 L CONSTS   0007 D COUNT
0002 D CR       005F L CRLF     00F6 L CTYPE    00ED L DESCND   0306 L ENDCK    0100 L ENDREC
0116 L ENT0     0301 L ENTEXIT  0100 L ENTRY    0404 L ERROR    0006 S F$EXIT   0334 L FLOOP
052A L FS1      0327 L FS2      0032 D GI       0240 L GET1     025F L GETKEY   0569 L GETREC
0580 L GETRET   0932 E NEEDED   0056 D I        0085 S I$CLOSE  0083 S I$CREA   0087 S I$DLET
0084 S I$OPEN   0088 S I$RDLN   0089 S I$READ   0088 S I$SEEK   008A S I$WRIT   008C S I$WRLN
04C3 L ILOOP    0539 L INC1     0040 D INDEX    0145 D INDISP   004F D IPATH    0034 D IRECL
0105 D INPUT    0056 D J        0058 D J1       0404 L JILOOP   0200 D JIREC    04CB L JLOOP
004E D JPATH    0100 D JREC     0044 D KEYS     0200 L KL1      0200 L KL2      0100 E KLASS
0700 L LIST     0049 L LIST1    003E D LKEY     003C D LKEY     0030 D LRECL    0003 D N1
0040 D MASTER   0050 D MPATH    0005 D MPBUFF   0391 L MPT      03A0 D MREC     0050 L MSDI
0063 L MS10     040D L NAME     08DE L NEWLN    03F1 L NOTEND   0061 L NP       08EC L NPAGE
0001 S OBJ      0155 D OTBUFF   0146 D OTDISP   0070 L OUTP     004A D OUTPUT   0370 L PARSE
03AA L PARSE1   0390 L PARSE2   0010 S PGM      005A D POS      0040 L PUTLN    012E L PTRNEL
058E L PUTREC   05B2 L PUTRET   0000 S R1       0687 L REBLD    0752 L REBLD1   076E L REBLD2
0061 D REC      0400 E RECMAX   00A3 D RECP     0080 S REENT    0005 D RESULT   0002 S REV
0002 L RL       0212 L RL1      023E L RL2      0728 L RLEOF    04AE L RLOOP    0000 D SEO
067F L SETCAR   014F L SIN1     0177 L SIN2     0007 L SK       0256 L SK1      0010 D SKEYS
0318 L SKIPOP   0400 L SORT     0040 D SORTBU   0181 L SOUT1    01F9 L SOUT2    0051 D SPATH
02EA L SRTB1    0740 D STACK    0740 E STKPTR   0900 E STOR     0500 L SWAP     0000 L SWOKE
0199 L SWORK1   0180 L SWORK2   0129 L TERM0    0137 L TERM1    0000 L TITLE    004C D TWORK
0042 D TYPE     02F0 L TYPE1    01A7 D WKDISP
```

"C" User Notes

Norm Commo
3 Pryor Road
Natick, MA 01760

On tap for this month are reviews of the Microware C compiler and the latest version of the Windrush compiler (which added floats and longs) and some errors that snuck into the last column.

ERRATA, OR... EATING CROW

None of us really likes to eat crow, and I'm no different; but it looks like I have to put on a bib, grab knife and fork, and dive in! The cause for all this misadventure is my own doing and revolves around I/O drivers publish in the last column.

First off, I was caught short by reader Pete Hayden. His letter pointed out that two of the example functions from the last column couldn't possibly work. The program had been compiled and run using the Windrush compiler. No problem there. ut as I started writing the column I thought it would be interesting to show that a "pointer to char" could be used to access hardware registers as well as an "array of chars." It being so simple, I just wrote out the examples. Enter the mistakes.

The first error was in the function serin() which used a pointer to char to address hardware. I fell flat by failing to POINT to the register in the while loop. That is, I left off the necessary indirection to really poll the status register. The correct version is written below.

```
    while (*port & 0x01 == 0)
        ;
```

and NOT

```
    while ((port & 0x01) == 0)
```

The first case will get the data from port and mask out all but the least significant bit. The one that appeared in the last column will take the address contained in port, and it with with bit 0 and see if the result was equal to zero. Hardly a good way to get data from an ACIA!

The error was repeated twice more in the very next example. The corrected lines, which appear on page 16, are

```
    while (*(port + STATUS) & RDRF  == NOTREADY)
        ;
```

and

```
    return *(port+DATA);
```

Making these changes will cure the ills you encountered if you actually implemented these version of the driver. My apologies for letting these slip in on you. As I said earlier, I had compiled and run the program and was sure that it would work; but my clay feet caught up with me when I "cheated" on some simple examples. So, hats off to Pete for being wide awake. And if anyone else catches me in an error don't be afraid to nail me. It's nice to know that the some of you know the language as good or better than me; and comforting too.

The next dilemma resulted from my (hopefully short term) inability to fulfill my plans for this column. We will not have an interrupt driven version of the program this time as predicted in the last column. I have a first cut compiled and running, but unreliably. It inevitably goes out to lunch after a while. I just haven't had the time to investigate the problem. Most likely, it's some sort of stack creep or stack corruption, but I can't see where or why from looking at the code.

The interrupt driven port is the ACIA input for the modem. The drivers were written in C. The console used FLEX's serial routines. Maybe that's the problem. I want to rewrite all the I/O, both the interrupt and polled in C, thereby short circuiting FLEX and see if that cures the problem.

I do not consider myself a FLEX expert and as a result may have left some critical memory location within FLEX improperly set up. Does anybody out there have any ideas? I've already tried using SBUG-E's IRQ ctor instead of FLEX's, but it still didn't work. I'll keep on trying. Only after it runs reliably will it get published.

You don't really want unreliable code. In fact, if you're at all like me, you're perfectly capable of generating lots of your own when you let down your guard.

WINDRUSH C COMPILER

I received the latest version of the Windrush C compiler. A lot has happened since the last version was released. They have added the float, do ble and long data types; the manual has been greatly improved; but best of all, they now include their own relocating assembler as part of the package.

Ron Anderson sent me a test program that he will be using to compare different types of compilers, including C compilers. It used some simple floating point operations. I compiled these with the Windrush compiler and had only one problem (which ROn Anderson had also run into). Although major, it appears easily fixable. Everything worked as it should once I reprogrammed to get around the bug. The functions exercised the float and double data types. I also added a test for longs. While I haven't done extensive testing, it seems to work well.

The problem is that a routine has apparently been left out of the standard library. One of the functions in the test program asks for a floating point number, then reads a string and calls atof() to convert the answer to a float. But when the program was linked, out popped an error message saying that addin was missing. Next I compiled a program that did nothing but try to call addin as a function. Still no luck. The result is that atof() in the standard library is unusable, as is scanf(). For the short term, the best thing to do if you have this compiler is to use the atof() from "The C Programming Language" that appears on page 69. I feel confident that Windrush will rectify the problem.

The new manual is well done. It has been organized in sections by topic (as it was before), but with each section printed on a different colored paper stock. This makes getting to a particular section, such as the library, very easy and fast.

While clearly written the manual is not a tutorial for the language. What you get describes this particular compiler along with its limitations and extensions; the support programs such as the relocating assembler and the linking loader; and the standard library. They have kept the very nice section on the compiler's error messages and their potential cause, which the new user should find quite helpful.

Finally, they have included their own relocating assembler. So now your money buys a functional package. Before, you had to have one of the TSC assemblers, preferably RELASM. That meant spending an additional $150 over the cost of the compiler package. Granted, the TSC's RELASM is worth the price, but it made the package expensive. Now all they need to add is a librarian and the package will be complete. If you already have TSC's LIB-GEN, you may use it with the Windrush assembler since the assembler's output format is TSC compatible.

One nice touch is the difference between using the angle brackets or the double quotes for including files. If you use the form

```
#include <filename>
```

the compiler will look for the file on the system disk; but if you use the form

```
#include "filename"
```

it will use the specific path name given between the quotes or apply the extant defaults. This means that you can keep your header files on the same disk as the compiler so you don't need them on all your program source disks.

Aside from the lack of a librarian, which I hope will come at a later I rate, this as AAA package.

MICROWARE C COMPILER

Now some good news for OS9 users who full C, the Microware C compiler is available. It was a long time coming, since Microware had some tough problems with their first C compiler consultant, but it seems to have been worth the wait.

The initial package offering includes both the Level 1 and Level 2 compilers, a relocating assembler, linkage editor, standard library and runtime support routines.

The compiler is just about the full implementation as defined in K&R. What's missing is bit fields and one or two of the fairly obscure compiler directives. You do get floats, doubles, longs, structs/unions and initializers. All the features that really make the language so nice.

The compiler is based on the McCosh compiler; but has been implemented and modified by Microware and is sold and supported only by them. Everything good that I've said about the Windrush compiler can be applied to Microware's compiler with the added advantage of having OS9 as the host operating system.

The package came with the compiler (for both levels 1 and 2), a relocating assembler, linker, standard library and runtime support module.

The manual is quite well done. Some of the major sections are

- the compiler's limitations and extensions
- using the compiler
- using the assembler
- the standard library functions
- compiler error messages
- interfacing with Basic09 modules

In all, I found the manual very easy reading. Like most compiler manuals, it is not a tutorial on the language, but it conveys what you need to know to put the compiler to good use.

Microware has added some interesting extensions to the language to make use of some OS9 characteristics. These are the data types direct, extern direct and static direct. When a variable is specified to be one of these types, it is allocated in the direct page region of memory for that program. This is a useful extension since it opens the possibilities of code shrinkage and higher execution speed. Of course, programs with this data types would suffer portability when moved to other compilers, which implies that it should be used judiciously.

Another extension is the ability to imbed assembler in the C source with the directives

```
#asm
    .
    .
    assembler code
    .
#endasm
```

You might just find this handy under certain conditions, or where speed is absolutely critical.

The compiler makes available a number of switches to the user for controlling the process. You may stop before assembly or linkage; specify no optimizing or stack checking; specify the name of the output file and a lot of other things. The switches available are

-a suppress assembly

-c include C source code as comments in the assembler code

-d<ident>=<string> define <ident> to be the value of <string>; equivalent to having done #define <ident> <string> from within the program

-e=<n> set the edition number to the value of <n>

-f=<path> specify the ouput of the linker to be <path>

-l=<filename> search library file <filename> during the link

-m=<size> specify the default memory size for stack, data and parameters

-o suppresses the optimizing phase

-p invokes the profiler option

-r suppresses the link phase

-s inhibits generation of stack checking code

The compiler assumes certain default file suffixes. They are

.c	C source file
.a	assembler source file
.r	assembled, relocatable module
none	an OS9 executable file

This combinations of assumed file suffixes and compiler switches gives the user great flexibility in controlling the compilation process. For example, suppose that you want to compile a single source module into an executable binary. You would type the command

```
cc filename.c
```

This would accomplish everything from compile to linkage and produce an execution module named "filename" on the system disk.

Now suppose that you had a program that consisted of a few modules and that you were debugging only module at the time. You could type the command line

```
cc file1.c file2.o file3.o file4.o
```

Here, the compiler would compile and assemble file1 and then link all four files together. The default name of the execution module would be "output" unless you overrode it with the -f=<path> switch.

Similarly, if you want to just assemble some previously compiled file you would type

```
cc filename.a
```

I've compiled a number of small programs with the compiler. I find it very user friendly with one or two exceptions.

The compiler is invoked by the command "cc1" or "cc" for levels 1 and 2 respectively. These programs merely build a shell script on the data directory called c-com; which then gets executed. This file calls in each pass of the compiler as well as the assembler and linker if required.

The problems appear to lie in cc1 (I have not tried cc extensively). The file suffixes must be in lower case, if they are in upper case the whole process stops. Secondly, when you use the -f option, you must specify the complete pathname. If you happen to say

```
cc filename.c -f=foo
```

the linker will not put the output file on the default data directory, it will put it on the default system. To get it put on anything other than the system directory requires that you give the complete pathname.

The output of the code is tight, although the seive program compiled ran about 3 seconds slower than when it was compiled with the Windrush compiler running under FLEX. This may have been due to OS9 system overhead but I'm not sure. It's not really that important. The code is tight and reasonably fast and that's what counts.

The standard library is pretty extensive. There is an index in the back of the manual of all the functions and what page they can be found on. I counted 117 functions in all. There are both high level and low level I/O routines and some nice support functions for generating 3 byte addresses used by OS9 I/O drivers. I've encountered no problems with the library to date.

I rate the package AAA. A lot of time and thought has gone into it and it shows. True, there are some minor aggrevations which I mentioned but I tend to overlook these. Maybe I shouldn't, it's just that I've never seen the "perfect" package. Probably couldn't afford to buy it if I did find it anyway.

Some good news for CoCo users who might read this column. Radio Shack will be selling OS9 for their new 64K CoCo and also available thru them will be Basic09, Pascal and most likely (at some later date) C.

WRAP UP

That's it for this column. Next time I hope to have a review of the Dynasoft compiler for OS9 along with a program that shows how command line arguments can be used effectively. A letter from Ron Anderson arrived yesterday, and it contained some interrupt driven input routines that he has used with editors. I will translate them into C code and see if that clears out the problems I've been having. Then we can get on track. Till then...

Compare Basics

Analyzing a Tower of Babel:
A Comparison of Eight Major
Versions of BASIC in Use on
the Motorola 6809 Computer

by E. M. (Bud) Pass, Ph.D.
Computer Systems Consultants, Inc.
1454 Latta Lane, Conyers GA 30207
Telephone Number 404-483-1717/4570

INTRODUCTION

Any user of multiple 6809 BASIC interpreters and compilers soon becomes painfully aware of the differences and incompatibilities among the BASIC processors available in the 6809. This article surveys eight of the BASICs in common use on 6809 systems, highlighting the major similarities and differences among them. These BASIC interpreters and compilers and their developers are as follows:

A	A/BASIC	Microware
C	Color BASIC	Microsoft
E	Extended Disk Color Basic	Microsoft
F	F/BASIC	Terminus Design
M	OS/9 BASIC09	Microware
T	BASIC	Technical Systems Consultants
U	UniFLEX Extended BASIC	Technical Systems Consultants
X	Extended BASIC	Technical Systems Consultants

The single-letter codes in the left column will occasionally be used as a shorthand to indicate the corresponding BASIC processor for the remainder of this article.

The comparison is organized into general considerations and syntax/semantics summary charts. These charts may also be used for references purposes for any of the BASICs in the table above.

COMPARISONS

MANUALS

On an external comparison basis, the first contact a user often has with a new language is the programming manual. Each of the BASICs in the table above has one manual, intended to be a combination training, reference, and user manual.

The best training manuals, by far, are those by Microsoft for their Color BASIC interpreters. These are printed in color, bound, and very attractive, especially to novices. They are divided by product into three manuals, as follows:

Color BASIC
Extended Color BASIC
Extended Disk Color BASIC

This breakdown is logical for several reasons. From Microsoft's viewpoint, it means that the manuals agree with the products on a one-to-one basis, making updates simpler to manage in the future. From the trainee's viewpoint, there is only one manual providing the information for the current module, since the products and manuals are cumulative, in the order provided above.

However, the Color BASIC manuals are not good reference manuals. They provide a table of contents, index, and quick-reference syntax and semantics charts, although there is some confusion about which verbs are in the various versions. The manuals are not oriented toward being used for reference, however. It is difficult or impossible to locate complete syntax, function, or verb information in many cases, as the manuals rely on examples and exercises to convey much of the language to the trainee.

The worst manual of the group is the one for A/BASIC. It has no table of contents nor index, provides no quick-reference syntax and semantics charts and, like most of the others, has no training section. There is a two-page sales flyer providing a summary of the language and an example program provided by Frank Hogg Lab. The manual has several references to apparently obsolete verbs, such as "OPEN FILES", which are not described in the manual.

The manual for F/BASIC is considerably better than the manual for A/BASIC. It has a table of contents and a quick-reference chart, although it is a reference manual and has no training section.

The verbs added to A/BASIC to produce F/BASIC which support the Terminus Design Arcade 50 graphics board are described quite well.

The manuals provided by Microware for BASIC09 and by TSC for their versions of BASIC are all adequate for reference purposes. They all have tables of contents and indices, but neither has a quick-reference verb chart. The BASIC09 manual provides a brief training section, but the TSC BASIC manuals do not.

PROCESSOR CATEGORIES

Of the processors listed above. A/BASIC and F/BASIC are the only true compilers; the others are all interpreters. All of the interpreters except the Color BASICs have a compressed program storage capability which some manuals refer to as "compilation", but is not, since machine language code is not produced. Compressed source programs load faster and are smaller than the original source programs, and offer some degree of security to software developers.

Each of the TSC interpreters is capable of "compiling" programs into a compressed form and saving it on disk for later use. TSC also offers several programs called "pre-compilers" which compress the BASIC source program into a form compatible with the BASIC interpreters. They offer the advantage of being able to process much larger programs, since they are smaller than the interpreters and never load the BASIC program into memory. They also offer extended language features, such as longer variable names and continued source lines.

BASIC09 is also capable of producing a compressed form, also known as "I-code" (for intermediate code). This form may be interpreted by either the BASIC09 program itself or by the BRUN program, which only provides run-time support and hence is smaller than BASIC09.

NUMBER REPRESENTATION

A/BASIC and F/BASIC are the only integer BASICs in the list above; all of the others also support floating point number representation. This is due to the fact that A/BASIC and F/BASIC are designed to be used in situations not requiring floating point operations, such as dedicated control and graphics applications. Since both A/BASIC and F/BASIC were based upon the 6800 version of A/BASIC, they are highly related. However, F/BASIC is not a true superset of A/BASIC, since each language has independently progressed from the source.

The following table presents the number representations supported by each of the BASIC processors discussed in this article:

A	16-bit integer
C	16-bit integer and 5-byte floating point
E	16-bit integer and 5-byte floating point
F	16-bit integer
M	8-bit and 16-bit integer and 5-byte floating point
T	16-bit integer and 4-byte floating point
U	16-bit integer and 8-byte floating point
X	16-bit integer and 8-byte floating point

The following table provides the precision available in each number representation in terms of decimal digits:

8-bit integer	2 digits (0 to 255)
16-bit integer	4 digits (-32768 to +32767 numeric, 0 to +65535 logical)
4 byte floating point	6 digits (approx. -10^+38 to -10^-38, 0, and +10^-38 to +10^+38)
5 byte floating point	9 digits (approx. -10^+38 to -10^-38, 0, and +10^-38 to +10^+38)
8 byte floating point	17 digits (approx. -10^+38 to -10^-38, 0, and +10^-38 to +10^+38)

Note that nine decimal digits of precision is only marginally sufficient for business data processing in the U.S. and is insufficient in many other countries. This implies that only Microware BASIC09 and TSC Extended BASIC provide enough -recision for business data processing and that exceptional care must be used with BASIC09 to avoid the loss of any precision whatsoever in calculations involving amounts of money.

Only BASIC09 and TSC Extended BASIC provide the means by which a programmer may explicitly specify the type of a numeric variable. BASIC09 does this thru the use of the following declarations:

```
     DIM .... :INTEGER
         and
     DIM .... :BYTE
```

which specify 16 and 8 bit integers, respectively. TSC Extended BASIC does this thru the use of a "%" immediately following a variable name to specify 16 bit integers.

Most of the BASIC processors described here provide the ability for a programmer to define a hexadecimal integer constant directly. A/BASIC, F/BASIC, and BASIC09 use a "$" to indicate that the following string of numbers and letters represents a hexadecimal constant. Extended Color BASIC uses "&H" to introduce a hexadecimal constant and "&O" or "&" to introduce an octal constant. The TSC BASICs use the function "HEX" to convert its string argument to an integer representing the hexadecimal string argument. Color BASIC is the only processor of the group without this ability.

VARIABLE NAMING AND SUBSCRIPTING

The A/BASIC and F/BASIC interpreters accept the variable name format of a letter optionally followed by a decimal digit for numeric variables and of a letter and "S" for strings. A string need not be declared before its use, in which case its maximum length is set to 32 characters, or it may be declared to have a length of up to 255 characters with a declaration of the following format:

```
     DIM A$(length)
```

Numeric and string arrays may be declared with fixed number of occurrences of up to 255 with declarations of the following formats:

```
     DIM A(occur)
     DIM A(occur1,occur2)
     DIM AS(occur,length)
```

Since the suffix "S" is logically part of the name, a numeric variable and a string may have the same name. However, the first element of an array is designated in A/BASIC and F/BASIC as number 1, and may be referenced with or without the subscript designation "(1)" or "(1,1)". Thus simple variables and arrays may not have the same names, or the simple variable will appear to reference the first element of the array. The BASE statement may be used to force specific memory allocation of arrays. This is the most restrictive set of rules of the group surveyed.

The Color BASIC interpreters accept the variable name format of a letter optionally followed by a string of mixed letters and digits, ending with "$" for strings. Only the first two characters of a variable name are significant. However, a numeric variable and a string variable may contain the same first two characters. Numeric and string arrays may be declared with fixed or variable number of occurrences with declarations of the following formats:

```
     DIM name(occur)
     DIM name(occur1,occur2)
     DIM name(occur1,occur2,occur3)
     DIM name$(occur)
     DIM nameS(occur1,occur2)
     DIM name$(occur1,occur2,occur3)
```

Since the suffix "$" is logically part of the name, a numeric variable and a string may have the same name. A simple variable may have the same name as an array, and yet logically refer to a different location in memory. The first element of an array is number 0.

The BASIC09 interpreter accepts the variable name format of a letter optionally followed by a string of mixed letters and digits and underlines. A string variable name may end with "$", in case no explicit declaration is required (although the maximum length will be set to 32 characters), or it may be explicitly declared with an arbitrary name and length with a declaration of the following format:

```
     DIM string_name[length]:STRING
```

Numeric and string arrays may be declared with fixed or variable number of occurrences with declarations of the following formats:

```
     DIM name(occur) ...
     DIM name(occur1,occur2) ...
     DIM name(occur1,occur2,occur3) ...
```

Since the suffix "$" is part of the name, a string name ending with "$" refers to a different memory location than any other variable name not ending with "$". Simple variables and arrays may not have the same name. The first element of an array is number 0 or 1, depending upon whether the "BASE 0" or "BASE 1" declarative is used.

The TSC BASIC interpreters accept the variable name format of a letter optionally followed by a letter or decimal digit for numeric variables and ended by a "$" for strings or "%" for integers.. The TSC pre-compilers extend this to accept the variable name format of a letter optionally followed by a string of letters, decimal digits, and underlines for numeric variables and ended by a "$" for strings, or '%' for integers. Numeric and string arrays may be declared with fixed or variable number of occurrences with declarations of the following formats:

```
     DIM name(occur)
     DIM name(occur1,occur2)
     DIM name%(occur)
```

```
DIM name%(occur1,occur2)
DIN name$(occur)
DIN name$(occur1,occur2)
```

Since the suffices "%" and "$" are part of the name, an integer variable, a floating point variable, and a string may have the same name. A simple variable may have the same name as an array, and yet logically refer to a different location in memory. The first element of an array is number 0.

OPERATOR PRECEDENCE

Operator precedence determines how certain incompletely parenthesized expressions are evaluated. For example, it determines whether the following expression:

```
1+2*3/4^5
```

should be logically grouped as follows:

```
(1+(2*(3/(4^5))))
```

or should be logically grouped as follows:

```
((((1+2)*3)/4)^5)
```

which is a less-commonly used but possible grouping.

Thus operator hierarchy is an important topic of comparison among the group of BASIC processors being considered here. Unfortunately, there are differences among the processors which may catch the unwery. If portability is a consideration, parenthesize questionable expressions to help avoid problems of different orders of evaluation and of different grouping.

A/BASIC and F/BASIC use the following order for operator precedence (high to low):

```
1.  (, )
2.  functions
3.  #, unary -
4.  &, !, %
5.  *, /
6.  +, -, string +
7.  =, <>, >, <, >=, <=, =>, =<
```

Operators on the same level are evaluated left to right.

The Microsoft and TSC BASICs use the following order for operator precedence (high to low):

```
1.  (, )
2.  functions
3.  ^
4.  unary -
5.  *, /
6.  +, -, string +
7.  =, <>, >, <, >=, <=
8.  NOT
9.  AND
10. OR
```

Operators on the same level are evaluated left to right, except for exponentiation ("^"), which is evaluated right to left.

BASIC09 uses the following order for operator precedence (high to low):

```
1.  (, )
2.  functions
3.  NOT, unary -
4.  ^, **
5.  *, /
6.  +, -, string +
7.  =, <>, >, <, >=, <=
8.  AND
9.  OR
10. XOR
```

Operators on the same level are evaluated left to right, except for exponentiation ("^"), which is evaluated right to left.

ERROR HANDLING

All of the BASIC processors except the Color BASICs use the same method for handling errors encountered during interpretation of the BASIC program. This method involves the "ON ERROR GOTO N" statement, which establishes an error handling routine starting with line "N". This statement may be used as many times as required to control the handling of errors.

When an error occurs, program control is transferred to line "N" and an error code number is placed into "ERR". This code number differs among the various classes of BASICs (A/F, M, T/U/X) and is the same within the classes of BASICs considered here. When the processing of the error situation is complete, the program either terminates, branches to a common point, or branches back to the offending line to retry the line which caused the error. The TSC BASICs help in this process in that they have the "RESUME" statement which branches back to the line which caused the error and they place this line number in "ERL".

The Color BASICs do not have the capability of intercepting any errors except end of file. This situation is checked for with the "EOF(n)" function, which is true if file "n" is in an end of file condition. A/BASIC, F/BASIC, and BASIC09 have a similar "EOF(#n)" function, in addition to their "ON ERROR" statements.

INPUT AND OUTPUT

Input and output are areas of great variation among the BASIC interpreters and compilers considered here. Only the key points of comparison will be listed below, as the differences are too great for brief discussion.

All of the BASICs have input and output for one terminal, one printer, and multiple disk filex. Some support multiple terminals and printers.

Extended Color BASIC, BASIC09, and the TSC Extended BASICs support formatted output with the "PRINT USING" option of the "PRINT" statement. The Color BASICs support terminal cursor control with the "PRINT @" option of the print statement.

All except A/BASIC and F/BASIC support prompting forms of the "INPUT" statement. All except A/BASIC, F/BASIC, and BASIC09 support a single character input function such as "INCH$(n)" or "INKEY$".

A/BASIC, Extended Color BASIC, BASIC09, and the TSC BASICs support random record-oriented disk input and output. The concepts used in Extended Color BASIC and the TSC BASICs are generally similar. However, the TSC BASICs also provide virtual arrays, which are actually record-oriented disk files, but with syntax almost identical to normal arrays.

LINES AND STATEMENTS

All of the BASIC processors surveyed here require that each line be numbered, in ascending order, except for BASIC09 and the TSC pre-compiler BASICs. The line numbers must all lie from 1 to 32767. BASIC09 and the TSC pre-compiler BASICs do not require each line to be numbered, although they may be; thus only those lines which are targets of GOTO or GOSUB statements need be numbered.

The TSC pre-compiler BASICs allow an alphanumeric label, along with a numeric label, starting in the first column. An alphanumeric label starts with a letter, and is optionally followed by a string of letters, digits, and underlines. The pre-compilers also allow statements to be continued from one line to the next thru the use of the "\"-carriage return sequence.

All of the BASICs allow multiple statements on the same line. A/BASIC, the Color BASICs, and F/BASIC use ":" to separate statements. BASIC09 uses "\" to separate statements. The TSC BASICs all allow either ":" or "\" to separate statements.

Not all the processors interpret multiple statements on the same line in the same manner. For example, A/BASIC and F/BASIC would interpret the following statements:

```
100 IF I=J THEN 300:GOTO 500
```

in a manner equivalent to the following statements in most other BASIC processors:

```
100 IF I=J THEN GOTO 300 ELSE GOTO 500
```

SYNTAX AND SEMANTICS CHARTS

The following tables present the syntax and semantics for all of the versions of BASIC described in this article. The tables are divided into functions, operators, and statements. The following notation is used in the tables:

e	numeric or string expresion
L	list of variables and string variables
n	numeric expression
N	numeric constant
S	statement
s	structure
s$	string expression
S$	string constant
v	variable
v$	string variable
x	numeric or Boolean expression
[ ]	optional
,...	repeated
type	BASIC09 variable type

The first column in the table contains letters representing the eight BASIC processors. The second column contains "fn" for functions, "op" for operators, and "st" for statements. The third column contains the name of the function, operator, or statement. The fourth column contains the parameters of the function, operator, or statement. One or more lines of semantics follow the syntax for each function, operator, or statement.

PROCESSOR	TYPE	NAME	PARAMETERS	SEMANTICS
ACEFMTUX	fn	ABS	(n)	absolute value of n
M	fn	ACS	(n)	erccosine of n
F	fn	ADC	(n)	A to D value of channel n
M	fn	ADOR	(structure)	address of argument
M	fn	ADDR	(v$[(n[,n])])	address of argument
M	fn	ADDR	(v[(n[,n])])	address of argument
U	fn	ARG$	(n)	command line parameter n
U	fn	ARGC%		command line parameter count
ACEFMTUX	fn	ASC	(s$)	numeric value of first character of s$
M	fn	ASN	(n)	arcsine of n
E MTUX	fn	ATN	(n)	erctangent of n
F	fn	BLACK		constant for black
F	fn	BLUE		constant for blue
A F	fn	BUF$		I/O buffer
ACEFMTUX	fn	CHR$	(n)	ASCII character corresponding to n
F	fn	CLEAR		constant for clear
U	fn	CLOCK$	[(n)]	current clock time or time at n
F	fn	COLOR	(n,n)	color of pixel at argument (n1,n2)
E MTUX	fn	COS	(n)	cosine of n
F	fn	CURX		x-coordinate of cursor
F	fn	CURY		y-coordinate of cursor
E	fn	CVN	(s$)	numeric equivelent of 5-byte s$
UX	fn	CVT$%	(s$)	integer equivalent of 2-byte s$
TUX	fn	CVT$F	(s$)	floating equivalent of 5/8-byte s$
UX	fn	CVT%$	(n)	2-byte equivalent of integer n
TUX	fn	CVTF$	(n)	5/8-byte equivalent of floating n
F	fn	CYAN		constant for cyan
MT X	fn	DATE$		current date and time
U	fn	DATE$	[(n)]	current date or date at n
F	fn	DBLUE		constant for dark blue
F	fn	DGREEN		constant for dark green
X	fn	DPEEK	(n)	16-bit numeric value at address n
F	fn	DRED		constant for dark red
F	fn	DYELLOW		constant for dark yellow
A FM	fn	EOF	(#n)	test file n for end of file condition
CE	fn	EOF	(n)	test file n for end of file condition
TUX	fn	ERL		error line number
A FMTUX	fn	ERR		error number
E MTUX	fn	EXP	(n)	e to power n
M	fn	FALSE		false
A F	fn	FILSIZ	(#n)	number of sectors in file n
E M	fn	FIX	(n)	rounded integer value of n
M	fn	FLOAT	(n)	floating equivalent of integer n
F	fn	FONT		address of character generator table
T X	fn	FRE	(n)	number of bytes of unallocated memory
F	fn	GREEN		constant for green
F	fn	GREY		constant for grey
TUX	fn	HEX	(s$)	numeric equivalent of hex string s$
E	fn	HEX$	(n)	hex string equivalent of numeric n
TUX	fn	INCH$	(n)	input one character from file n
CE	fn	INKEY$		input one character from keyboard
TUX	fn	INSTR	(n,s$,s$)	first occurrence of s2$ in s1$ starting at character n (zero if not found)
E	fn	INSTR	([n,]s$,s$)	first occurrence of s2$ in s1$ starting at character n or 1 (zero if not found)
CE MTUX	fn	INT	(n)	truncated integer value of n
CE	fn	JOYSTK	(n)	A to D value of joystick designated by n
F	fn	JSWITCH	[(n)]	state of one or all joystick switches
M	fn	LAND	(n,n)	logical and of n1 and n2
ACEFMTUX	fn	LEFT$	(s$,n)	string representing n characters starting at left of s$
ACEFMTUX	fn	LEN	(s$)	length in bytes of s$
F	fn	LGREEN		constant for light green
M	fn	LNOT	(n)	logical not of n
E MTUX	fn	LOG	(n)	natural logarithm of n
M	fn	LOG10	(n)	base-10 logarithm of n
M	fn	LOR	(n,n)	logical inclusive or of n1 and n2
F	fn	LRED		constant for light red
M	fn	LXOR	(n,n)	logical exclusive or of n1 and n2
F	fn	MAGENTA		constant for magenta
CE	fn	MEM		number of bytes of unallocated memory
U	fn	MEM	(n)	number of bytes of allocated memory
ACEFMTUX	fn	MID$	(s$,n[,n])	string representing n2 (or remaining) characters starting at n1 characters into s$
E	fn	MKN$	(n)	5-byte equivalent of numeric n
F	fn	ORGX	(n)	x-coordinate of sprite on plane n
F	fn	ORGY	(n)	y-coordinate of sprite on plane n
ACEFMT X	fn	PEEK	(n)	8-bit numeric value at address n
MTUX	fn	PI		pi (3.14159265)
F	fn	PIXEL	(n,n)	test pixel at argument coordinate (n1,n2)
CE	fn	POINT	(n,n)	test graphics cell at argument coordinate (n1,n2)
A FM	fn	POS		character position in print buffer
E TUX	fn	POS	(n)	character position in file n buffer
E	fn	PPOINT	(n,n)	color of cell at argument coordinate (n1,n2)
T X	fn	PTR	(v$[(n[,n])])	address of argument
T X	fn	PTR	(v[(n[,n])])	address of argument
F	fn	RED		constant for red
ACEFMTUX	fn	RIGHT$	(s$,n)	string representing n characters at at the end of s$
CE MTUX	fn	RND	(n)	random number between 0 and 1
A	fn	RND	[(n)]	random number between 0 and 32767
F	fn	RND	[(n)]	random number between 0 and 255
F	fn	SCOLOR	(n)	color of argument sprite plane n
U	fn	SECOND		seconds since midnight 1/1/1980
CE MTUX	fn	SGN	(n)	sign of n

```
 CE MTUX    fn  SIN     (n)
                              sine of n
    M       fn  SIZE    (structure)
                              size in bytes of argument
    M       fn  SIZE    (v$[(n[,n])])
                              size in bytes of argument
    M       fn  SIZE    (v[(n[,n])])
                              size in bytes of argument
    F       fn  SNAME   (n)
                              sprite image name of sprite plane n
     TUX    fn  SPC     (n)
                              generate n spaces
    M       fn  SQ      (n)
                              square of n
  E MTUX    fn  SQR     (n)
                              square root of n
    M       fn  SQRT    (n)
                              square root of n
A   F       fn  STATUS  (#n)
                              status of file n
A EFMTUX    fn  STR$    (n)
                              string conversion of numeric n
  E    U    fn  STRING$ (s$,n)
                              string composed of n occurrences of s$
A   FM      fn  SUBSTR  (s$,s$)
                              first occurrence of s1$ in s2$
                              (or zero if not found)
A   F       fn  SWAP    (n)
                              swap bytes of 16-bit value n
    F       fn  SWITCH  [(n)]
                              test 8-bit input port
ACEFMTUX    fn  TAB     (n)
                              advance print buffer pointer to
                              position n
  E MTUX    fn  TAN     (n)
                              tangent of n
       U    fn  TASK%
                              task number
       U    fn  TERM%
                              terminal number
       U    fn  TIME$   [(n)]
                              current date/time or date/time of n
    M       fn  TRIM$   (s$)
                              argument s$ without trailing spaces
A   F       fn  TRM$    (s$)
                              argument s$ without trailing spaces
    M       fn  TRUE
                              true
       U    fn  TSTAT%
                              task termination status
      T X   fn  USR     (n)
                              user function with parameter n
  E         fn  USRn    (n)
                              user function n1 with parameter n2
ACEFMTUX    fn  VAL     (s$)
                              numeric conversion of string s$
  E         fn  VARPTR  (v$[(n[,n])])
                              address of argument
  E         fn  VARPTR  (v[(n[,n])])
                              address of argument
    F       fn  VPEEK   (n)
                              8-bit numeric value at vram address n
    F       fn  WHITE
                              constant for white
    F       fn  YELLOW
                              constant for yellow
A   F       op  !
                              binary logical inclusive or (l!r)
ACEFMYUX    op  "           ... "
                              unary string constant definition
A   F       op  #
                              unary logical negation (#r)
A   FM      op  $
                              unary hexadecimal constant definition
A   F       op  %
                              binary logical exclusive or (l%r)
A   F       op  &
                              binary logical and (l&r)
  E         op  &H
                              unary hexadecimal constant definition
  E         op  &[O]
                              unary octal constant definition
     TUX    op  '           ... '
                              unary string constant definition
ACEFMTUX    op  (
                              unary expression group start
A   F       op  (#
                              unary file number group start
ACEFMTUX    op  )
                              unary expression group end
ACEFMTUX    op  *
                              binary numeric multiply (l*r)
    M       op  **
                              binary numeric exponentiation (l**r)
ACEFMTUX    op  +
                              binary numeric addition (l+r),
                              binary string concatenation (l$+r$),
```

```
                              unary numeric positive (+r)
ACEFMTUX    op  ,
                              binary subscript separator,
                              binary parameter separator,
                              binary PRINT punctuation (tab),
                              unary PRINT punctuation (CRLF)
ACEFMTUX    op  -
                              binary numeric subtraction (l-r),
                              unary numeric negation (-r)
 CE MTUX    op  .
                              decimal point
ACEFMTUX    op  /
                              binary numeric division (l/r)
ACEF TUX    op  :
                              binary separate statements on line
    M       op  :
                              binary type assignment (variable:type)
    M       op  :=
                              binary numeric assignment (l:=r),
                              binary string assignment (l$:=r$)
ACEFMTUX    op  ;
                              binary PRINT punctuation (no tab),
                              unary PRINT punctuation (no CRLF)
ACEFMTUX    op  <
                              binary numeric less (l<r),
                              binary string less (l$<r$)
ACEFMTUX    op  <=
                              binary numeric not greater (l<=r),
                              binary string not greater (l$<=r$)
ACEFMTUX    op  <>
                              binary numeric not equal (l<>r),
                              binary string not equal (l$<>r$)
ACEFMTUX    op  =
                              binary numeric equal (l=r),
                              binary string equal (l$=r$)
                              binary numeric assignment (l=r),
                              binary string assignment (l$=r$)
ACEFM       op  =<
                              binary numeric not greater (l=<r),
                              binary string not greater (l$=<r$)
ACEFM       op  =>
                              binary numeric not less (l=>r),
                              binary string not less (l$=>r$)
ACEFMTUX    op  >
                              binary numeric greater (l>r),
                              binary string greater (l$>r$)
ACEFMTUX    op  >=
                              binary numeric not less (l>=r),
                              binary string not less (l$>=r$)
 CE MTUX    op  AND
                              binary logical and (l AND r)
 CE MTUX    op  NOT
                              unary logical negation (NOT r)
 CE MTUX    op  OR
                              binary logical inclusive or (l OR r)
    M       op  XOR
                              binary logical exclusive or (l XOR r)
    MTUX    op  \
                              binary separate statements on line
  E MTUX    op  ^
                              binary numeric exponentiation (l^r)
       U    op  ~
                              binary numeric approximately equal (l~r)
 CE         st  '           ...
                              introduce remark
    M       st  (*          ... *)
                              introduce remark
A   F       st  *           ...
                              introduce remark (column 1)
A   F       st  .           ...
                              introduce remark (column 1)
  E         st  ?           [#n[,]][@n[,]][USING s$[,]][e[,]][e[;]][,...]
                              output characters to file n1 or to
                              terminal after setting cursor to n2
 C          st  ?           [#n[,]][@n[,]][e[,]][e[;]][,...]
                              output characters to file n1 or to
                              terminal after setting cursor to n2
 CE         st  AUDIO    OFF
                              disconnect cassette output from TV
 CE         st  AUDIO    ON
                              connect cassette output to TV
    F       st  BACKDROP[=]n
                              set color of backdrop plane to n
    M       st  BASE     0
                              set array subscript offset to 0
    M       st  BASE     1
                              set array subscript offset to 1
A   F       st  BASE     [=]n
                              set ram assignment address to n
    F       st  BCOLOR   [=]n
                              set background table color to n
    F       st  BLANK    [=]n
                              set VDP bit to blank or enable screen
    M       st  BYE
                              terminate program
A   F       st  CALL     n
                              call machine language subroutine
                              at address n
```

A FM	st	CHAIN	s$	load and run BASIC program named s$
TUX	st	CHAIN	s$ [N]	load and run BASIC program named s$ starting with line N or first line
M U	st	CHD	s$	set default data directory to s$
M	st	CHX	s$	set default execution directory to s$
E	st	CIRCLE	(n,n),n[,[n][,[n][,[n][,[n]]]]]	draw circle with center at (n1,n2), radius n3, color n4, height/width ratio n5, start point n6, end point n7
CE	st	CLEAR	[n[,n]]	reserve n1 bytes for string storege space, specify highest BASIC address n2
A FM	st	CLOSE	#n[,#n[,...]]	close specified files n1, n2, ...
A F	st	CLOSE	FILES	close all files
CE	st	CLOSE	[#]n[,[#]n[,...]]	close specified files n1, n2, ...
TUX	st	CLOSE	n[,n[,...]]	close specified files n1, n2, ...
CE	st	CLS	[n]	clear display to color n or to green
E	st	COLOR	[n[,n]]	set foreground color to n1 and sat background color to n2
U	st	COMMON	n	reserve n bytes for common storage
A	st	CREATE	#n,s$	create fila n with name s$
M	st	CREATE	#n,s$[:][UPDATE][[+]WRITE]	create file n with name s$
CE MTUX	st	DATA	N[,N[,...]]	establish constant table within program
CE MTUX	st	DATA	S$[,S$[,...]]	establish constant table within program
TUX	st	DEF	FNv(v)=n	define numeric function FNv1 with parametar v2
E	st	DEF	FNv(v,v,...)=n	define numeric function FNv1 with parameters v2, v3, ...
E	st	DEFUSRn	=n	set user function n1 entry point to n2
M	st	DEG		assume angles stated in degrees
F	st	DELAY	n	delay execution of program n/60 seconds
M	st	DELETE	s$	delete file named s$
TUX	st	DIGITS	n[,n]	set number of digits to be printed
CE T	st	DIM	v$(n[,n])[,...]	declare dimensioned variables
ACEF T	st	DIM	v(n[,n])[,...]	declare dimensioned variables
A F	st	DIM	v$(n[,n])[,...]	declare string variables with length n1 or string arreys of dimension n1, length n2
TUX	st	DIM	[#n,]v$(n[,n])[,...][=n]	declare dimensionad variables or declare virtual array associated with fila n1
TUX	st	DIM	[#n,]v(n[,n])[,...]	declare dimensioned variables or declare virtual erray essociated with file n1
M	st	DIM	v$[(n[,n])][,...][:type]	declare and allocete space for verisbles
M	st	DIM	v[(n[,n])][,...][:type]	declare and allocate space for variables and optionally specify types
A F	st	DISPLAT	[e[,]][e[;]][,...]	output strings and control characters to terminal
X	st	DPOKE	n,n	store 16-bit value n2 at address n1
F	st	DRAW	n,n[,n,n][,...]	draw line from cursor to (n1,n2) to (n3,n4) to ...
E	st	DRAW	s$	drew line as specified by s$
ACEF TUX	st	END		terminate execution
M	st	END	[L]	print list L (if present) end return to calling module
F	st	ENDEF		end sprite or pattern definition
M	st	ENDEXIT		provide end point for EXITIF
M	st	ENDIF		provide end point for IF
M	st	ENDLOOP		provide end point for LOOP
M	st	ENDWHILE		provide end point for WHILE
F	st	ENVL	n,n[,n]	set PSG number to n1, shape to n2, period to n3
M	st	ERROR	(n)	generete error n
TUX	st	EXEC	,s$	send command line s$ to operating system
CE	st	EXEC	[n]	execute machine language program at address n or last transfer address
U	st	EXIT	[N]	terminates program with coda n (or zero)
M	st	EXITIF	x THEN S	provide condition for terminating loop
A	st	EXPAND		expand memory space
F	st	FCOLOR	[=]n	set foreground table color to n
E TUX	st	FIELD	#n,n AS v$[(n[,n])][,...]	establish fields in random I/O buffer
U	st	FIELD	common,n AS v$[(n[,n])][,...]	establish fields in common sres
E	st	FILES	[n[,n]]	reserve n1 file buffers of total n2 bytes
ACEFMTUX	st	FOR	v=n TO n [STEP n]	create loop with control variable v set initially to n1, terminal condition of v crossing n2, step size n3 (or 1)
A F	st	GEN	N[,N[,...]]	insert specified values into program
M	st	GET	#n,structure	read random file n1 record
TUX	st	GET	#n[,RECORD n]	read random file n1 record n2 or next
E	st	GET	#n[,n]	read random file n1 record n2 or next
E	st	GET	[(n,n)]-[(n,n)],v[,G]	read graphic contents of rectangle into erray v
ACEFMTUX	st	GOSUB	N	call subroutine starting at line N
ACEFMTUX	st	GOTO	N	branch to line N
A F	st	IF	x GOSUB N	call subroutine starting et line N if expression x true
CE TUX	st	IF	x GOTO N	branch to line N if expression x true
A FM	st	IF	x THEN N	branch to line N if espression x true
CE TUX	st	IF	x THEN N [ELSE N]	branch to line N1 if expression x true; otherwise branch to line N2
ACE MTUX	st	IF	x THEN S [ELSE S]	perform stateent S1 if expression x true; otherwise perform statement S2
F	st	INIT	[=]n	set base address of ARCADE board to n
A F	st	INPUT	L	input list L from terminal
TUX	st	INPUT	LINE [#n,]v$	input string v$ from file n or terminal
CE TUX	st	INPUT	[#n,][S$;]L	issue prompt S$ to file n or terminal and input list L from file n or terminal
M	st	INPUT	[#n,][S$[,]]L	issue prompt S$ to fila n or terminal and input list L from file n or terminal
M	st	KILL	*	unlink all external procedures
M	st	KILL	[s$[,s$[,...]]]	unlink esternal procedures s1$, s2$, ...
A EF TUX	st	KILL	s$	delete file named s$
A F	st	[LET]	BUF$=s$	replace contents of I/O buffer with string expression to right of equal
E	st	[LET]	MID$(v$[(n[,n])],n[,n])=s$	replace specified portion of string variable in MID$ with string expression to right of equal
M	st	[LET]	structure[:]=structure$	assign expression on right of equal or colon-equal to variable on left
A F	st	[LET]	v$[(n)]=s$	assign expreesion on right of equal to variable on left
CE TUX	st	[LET]	v$[(n[,n])]=s$	assign expression on right of equal to variable on left
M	st	[LET]	v$[(n[,n])][:]=s$	assign expression on right of equal or colon-equal to variable on left
ACEF TUX	st	[LET]	v[(n[,n])]=n	

```
                                    assign expression on right of equal
                                    to variable on left
    M        st  [LET]    v[(n[,n])][:]=n
                                    assign expression on right of equal
                                    or colon-equal to variable on left
  E          st  LINE     INPUT[#n,][S$;]v$
                                    issue prompt S$ to file n or terminal,
                                    input string v$ from file n or terminal
  E          st  LINE     [(n,n)][-[(n,n)]][,PSET][,PRESET][,B[F]]
                                    draw line from start point to end point
    M        st  LOOP
                                    initiate program loop
  E  TUX     st  LSET     v$[(n[,n])]=s$
                                    assign expression on right of equal
                                    to field variable on left of equal
                                    in left-justified mode
   F         st  MAG      [=]n
                                    set sprite magnification factor to n
   F         st  MODE     [=]n
                                    set TMS-9918A mode to n
 CE          st  MOTOR    OFF
                                    turn cassette motor off
 CE          st  MOTOR    ON
                                    turn cassette motor on
   F         st  MOVE     n,n
                                    set invisible graphics cursor to (n1,n2)
A            st  NAME     S$
                                    set module name to S$
ACEFMTUX     st  NEXT     v
                                    initiate next iteration for FOR loop
                                    with control variable v
A  FMTUX     st  ON       ERROR GOTO [N]
                                    set trap at line N for error handling
                                    or terminate error handling trap
    M        st  ON       ERROR [GOTO N]
                                    set trap at line N for error handling
                                    or terminate error handling trap
A  F         st  ON       NOVR GOTO N
                                    branch to line N if no overflow
A  F         st  ON       OVR GOTO N
                                    branch to line N if overflow
ACEFMTUX     st  ON       n GOSUB N[,N[,...]]
                                    call subroutine at n-th line number N
ACEFMTUX     st  ON       n GOTO N[,N[,...]]
                                    branch to n-th line number N
A  F         st  OPEN     #n,s$
                                    open file n1 with name s$
    M        st  OPEN     #n,s$[:[READ][+WRITE][+UPDATE][+EXEC][+DIR]]
                                    open file n1 with name s$ and attributes
     T X     st  OPEN     NEW s$ AS n
                                    open new file n1 with name s$
      U      st  OPEN     NEW s$ AS n[,SIZE n]
                                    open new file n1 with name s$
  E          st  OPEN     S$,[#]n,s$[,n]
                                    open file n1 with name s$
 C           st  OPEN     S$,n,s$
                                    open file n1 with name s$
     T X     st  OPEN     [OLD] s$ AS n
                                    open file n1 with name s$
      U      st  OPEN     [OLD] s$ AS n[,SIZE n]
                                    open file n1 with name s$
A  F         st  OPT      L
                                    provide compile options
A  F         st  ORG      [=]n
                                    set program address to n
A  F         st  PAG
                                    continue compiler listing on next page
  E          st  PAINT    (n,n),n,n
                                    paint graphic screen from (n1,n2)
                                    with color n3 and border n4
    M        st  PARAM    v$[(n[,n])][,...][:type]
                                    declare space for variables
                                    and optionally specify types
    M        st  PARAM    v[(n[,n])][,...][:type]
                                    declare space for variables
                                    and optionally specify types
   F         st  PATDEF   n,N
                                    define pattern at vram address
                                    n of length N bytes
    M        st  PAUSE    [L]
                                    print list L (if present) and
                                    suspend execution and enter debug mode
  E          st  PCLEAR   n
                                    reserve n graphic pages
  E          st  PCLS     [n]
                                    clear screen to color n or
                                    to background color
  E          st  PCOPY    n TO n
                                    copy graphics from page n1 to page n2
   F         st  PIXMODE  [=]n
                                    modify single pixel generation
  E          st  PLAY     s$
                                    play music as specified by s$
  E          st  PMODE    [n[,n]]
                                    select resolution n1 and page n2
A  F         st  POKE     (n)=n
                                    store 8-bit value n2 at address n1
```

```
CEFMT X      st  POKE     n,n
                                    store 8-bit value n2 at address n1
      U      st  POSITION#n,n[,MODE n][,RESPONSE v]
                                    position sequential file n1 to byte
                                    n2 with mode n3 and response n4
  E          st  PRESET   (n,n)
                                    reset point at (n1,n2)
  E          st  PRINT    [#n[,]][@n[,]][USING s$[,]][e[,]][e[;]][,...]
                                    output characters to file n1 or to
                                    terminal after setting cursor to n2
 C           st  PRINT    [#n[,]][@n[,]][e[,]][e[;]][,...]
                                    output characters to file n1 or to
                                    terminal after setting cursor to n2
    M UX     st  PRINT    [#n[,]][USING s$[,]][e[,]][e[;]][,...]
                                    output characters to file n or terminal
     T       st  PRINT    [#n[,]][e[,]][e[;]][,...]
                                    output characters to file n or terminal
A  F         st  PRINT    [e[,]][e[;]][,...]
                                    output characters to terminal
    M        st  PROCEDURE S$
                                    introduce module named S$
  E          st  PSET     (n,n[,n])
                                    set point at (n1,n2)
   F         st  PSG      n,n,n
                                    set PSG n1 register n2 to n3
    M        st  PUT      #n,structure
                                    write random file n1 record
     TUX     st  PUT      #n[,RECORD n]
                                    write random file n1 record n2 or next
  E          st  PUT      #n[,n]
                                    write random file n1 record n2 or next
  E          st  PUT      [(n,n)]-[(n,n)],v[,PSET][,PRESET][,AND][,OR][,NOT]
                                    set graphic contents of rectangle
                                    from array v
    M        st  RAD
                                    assume angles stated in radians
   F         st  RDRAW    n,n[,n,n][,...]
                                    draw line from cursor to relative
                                    position (n1,n2) to relative
                                    position (n3,n4) ...
A  F         st  READ     #n,L
                                    read data into list L from
                                    sequential file n
 CE MTUX     st  READ     L
                                    read data into list L from
                                    DATA statements
ACEFMTUX     st  REM
                                    introduce remark
  E          st  RENAME   s$ TO s$
                                    rename file named s1$ as s2$
     TUX     st  RENAME   s$,s$
                                    rename file named s1$ as s2$
    M        st  REPEAT
                                    introduce REPEAT ... UNTIL statement
  E          st  RESET    (n,n)
                                    reset point at (n1,n2)
 CE          st  RESTORE
                                    reset DATA pointer to first statement
A  F         st  RESTORE #n[,#n[,...]]
                                    rewind files n1, n2, ... and
                                    reopen for input
    MTUX     st  RESTORE [N]
                                    reset DATA pointer to first statement
                                    or to statement on line N
     TUX     st  RESUME   [N]
                                    return to program from error routine
                                    to original line or to line N
ACEFMTUX     st  RETURN
                                    return from most recent active GOSUB
   F         st  RMOVE    n,n
                                    set invisible graphics cursor to (n1,n2)
                                    relative to current cursor
A            st  RREAD    #n,n,L
                                    read data into list L from
                                    random file n1 record n2
  E  TUX     st  RSET     v$[(n[,n])]=s$
                                    assign expression on right of equal
                                    to field variable on left of equal
                                    in right-justified mode
   F         st  RSPRITE n,n,n[,n,n]
                                    move sprite image on plane n1 to
                                    (n2,n3) relative to current sprite
                                    image on that plane and may specify
                                    new sprite name n4 and color n5
    M        st  RUN      s$[(e[,e[,...]])]
                                    call module named s$ passing
                                    parameters e1, e2, ...
A            st  RWRITE   #n,n,L
                                    write data from list L into
                                    random file n1 record n2
   F         st  SCALE    n,n
                                    specify RND lower bound n1 and
                                    upper bound n2
A  F         st  SCRATCH #n[,#n[,...]]
                                    rewind files n1, n2, ... and
                                    reopen for output
  E          st  SCREEN   n,n
```

```
                                  select graphics mode as specified by
                                  n1 and color set as specified by n2
    M        st  SEEK    #n,n
                                  position random file n1 to byte n2
      U      st  SEEK    #n,n[,MODE n][,RESPONSE v]
                                  position random file n1 to byte
                                  n2 with mode n3 and response n4
 CE          st  SET     (n,n,n)
                                  set point at (n1,n2) to color n3
   F         st  SET     n,n[,n,n][,...]
                                  modify pixels at (n1,n2), (n3,n4)
                                  as specified by most recent PIXMODE
A   M        st  SHELL   s$
                                  pass message s$ to operating system
   F         st  SIZE    [=]n
                                  specify number of pixels in sprite
                                  image from value of n
      U      st  SLEEP   n
                                  delay execution of program n seconds
 CE          st  SOUND   n,n
                                  sound tone specified by n1 for
                                  n2 units of 6/100 second duration
   F         st  SPCOLOR n,n
                                  set color of sprite on plane n1 to n2
   F         st  SPDEF   n,N
                                  define sprite image at vram address
                                  n of length N bytes
   F         st  SPNAME  n,n
                                  display sprite n2 on plane n1
   F         st  SPRITE  n,n,n[,n,n]
                                  move sprite image on plane n1 to
                                  (n2,n3) and may specify new sprite
                                  name n4 and color n5
A  F         st  STACK   [=]n
                                  set initial stack pointer to address n
ACEF TUX     st  STOP
                                  terminate program execution
    M        st  STOP    [L]
                                  print list L (if present) and
                                  terminate execution of program
 CE  TUX     st  SWAP    v$[(n[,n])],v$[(n[,n])]
                                  exchange contents of specified
                                  variables
 CE  TUX     st  SWAP    v[(n[,n])],v[(n[,n])]
                                  exchange contents of specified
                                  variables
  E          st  TIMER   =n
                                  set timer to n
   F         st  TONE    n,n[,n]
                                  set channel n1 frequency n2 and
                                  amplitude n3
  E M        st  TROFF
                                  turn off program trace
  E M        st  TRON
                                  turn on program trace
    M        st  TYPE    type[:]=structure
                                  establish structure type
      U      st  UNLOCK  n[,n[,...]]
                                  release interlocks on files n1, n2, ...
    M        st  UNTIL   x
                                  provide termination condition x
                                  for REPEAT ... UNTIL loop
   F         st  VPOKE   n,n
                                  store 8-bit value n2 at vram address n1
   F         st  VPRINT  [e[,]][e[;]][,...]
                                  place characters on pattern plane
   F         st  VREG    n,n
                                  initialize TMS-9918A register n1 to n2
      U      st  WAIT    n
                                  set time limit for input to n seconds
    M        st  WHILE   x DO
                                  introduce and provide termination
                                  condition x for WHILE ... ENDWHILE loop
      U      st  WIDTH   n
                                  set line width to n for PRINT
ACEFM        st  WRITE   #n,L
                                  write data from list L into
                                  sequential file n
      U      st  {       ... }
                                  introduce remark
```

SUPPORT YOUR ADVERTISERS

BIT BUCKET

COMPUTER SYSTEMS CONSULTANTS, INC.
1454 LATTA LANE
CONYERS, GEORGIA, 30207
TELEPHONE 404-483-1717 OR 4570

CSC MAILING LIST SYSTEM IMPROVEMENT N TICE NUMBER I

The CSC Mailing List System has recently been extensively modified. The major change is the modification of the indexing scheme to provide a very significant decrease in the time required to find a record on the file. You can receive a copy of the revised system by returning the original disk to the address above. If you purchased the Mailing List System over one year ago, please include $5.00 to cover the costs of shipping and a new manual. If you cannot find the original disk or do not wish to return it, send proof of purchase and $10.00 for a new disk and manual.

Thank You,

Bud

E. M. (Bud) Pass

St Charles City County Library

6-1-83

Contact: Carl H. Sandstedt, Director
St. Charles City-County Library
P.O. Box 529
St. Peters, MO 63376
(314) 441-2300

Release Date IMMEDIATE

NEWS RELEASE

AFFORDABLE LIBRARY AUTOMATION

The St. Charles City-County Library District has chosen "LISTEN" marketed by B.E.P. Associates of Clayton, Missouri for its complete automation system. Library Director Carl Sandstedt has worked closely with the President of B.E.P., Larry Bleiberg in customizing the automation system that may be implemented in phases by building on the original system with additional hardware and software, that addresses the problems of a multi-branch county-wide library district, and that is affordable. Bleiberg, who is also a librarian, has unique insights into the needs and problems of libraries. The LISTEN system which is written in ANSI standard MUMPS and normally runs on a HELIX 6809 based microcomputer, will automate the overdues, acquisitions, and circulation systems and provide cataloging for a fiche catalog which will eventually develop into an on-line catalog.

Phase I of automation is already in progress. A specially designed subsystem is replacing a costly, time consuming and inaccurate manual system of notifying patrons of overdue materials. Implementation of automated overdues provides a more effective means of retrieving overdue materials and fines and allows the District to be stricter with abusers of library service. Hardware and software for this phase which will run less than $20,000 should pay for itself in the first year through savings in staff time. The additional staff time will be used to provide better public service. A unique feature of the overdues system is that notices will be transmitted electronically to the Post Office using their E-Com system which will print, stuff and mail a one-page overdue notice for $.26 (includes postage)...a considerable savings from in-house mailings.

The second step of Phase I will automate materials acquisition by utilizing a BISAC standard which allows electronic transmission of orders to book jobbers. The St. Charles City-County Librar D istrict will be one of the first in the nation with full BISAC capability.

Future phases of LISTEN will include the fully automated circulation system which will be an extension of the original overdues system, and full MARC bibliographic records which will be condensed to form a fiche catalog. Eventually the public catalog will be completely on-line.

Hardware for LISTEN includes a Helix computer as the basic component. The Helix is manufactured by Hazelwood Computers, Inc. in O'Fallon, Missouri. When configured with a Motorola 68000 processor, the Helix will address up to sixteen megabytes in memory. Disc capacity for the system, originally at forty megabytes, will eventually expand to three hundred megabytes or greater. Although it is doubtful that any microcomputer can stand the rigors of servicing multiple branches for online catalog usage and circulation, the District plans to install separate CPU's in each of its major branches. Even with replicating computers in multiple locations, total cost of LISTEN is expected to be only a fraction of minicomputer based circulation systems. Sandstedt, who has seen the District undertake a major building program while cutting property taxes 10%, sees a parallel between LISTEN and his new buildings. "We have always prided ourselves with our ability to build new libraries at a cost of $28 to $32 dollars per square foot and LISTEN's the same kind of project. We feel that we can get a lot more for the taxpayer's money than we could with the typical turnkey system."

AFFORDABLE LIBRARY AUTOMATION

St. Charles City-County Library is the first public library using LISTEN. The District circulated slightly more than a million books and other materials in 1982 and has an operating budget of approximately $1.5 million. The District serves a population of slightly more than 150,000 and is located adjacent to a major urban area. LISTEN offers an automation system that will expand with a growing library district and is also adaptable and affordable for a very small library.

ELEKTRA Means Quality

ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 16-1/2" wide by 21-7/8" deep by 8-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for D type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI-Abort) and two cutouts for 5-1/4" disk drives. $250.00

Filler plate for 5-1/4" drive opening $10.00 Fan Filter $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. 3 primary input for light rated and heavy loading.
220v Version $200.00 110v Version $175.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drives $50.00
Heavy duty version for 1 Winchester drive and 1 floppy dr $75.00

ELEKTRA UNIVERSAL MOTHERBOARD Heavyweight 0.125" thick, 18" long by 9" wide. 11 memory (50 pin) slots. 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively. 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base address for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.
Bareboard w/documentation $80.00
Kit w/gold connectors $320.00 Assembled w/gold connectors $380.00
Kit with connectors $240.00 Assembled with connectors $300.00

ELEKTRA CHASSIS Includes cabi l, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested $850.00

ELECTRA CPU 8/9 Use either the 6802 or 6808 (to run 6800 software) or 6809. Has provision for up to 3 2716 EProms, 1K scratch pad, MC6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and SBUG-E, HUMBUG, and MICROBUG in the 6809 mode. Versions of OS-9™ Level I are available.
Bareboard $50.00 Kit $225.00 Assembled $275.00
2K 6800 HUMBUG $40.00 2K 6809 MICROBUG $30.00 See 4K HUMBUGS next page
Optional Baud Rate Generator $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRQ or FIRQ/NMI bus line.
Bareboard $20.00 Kit $60.00 Assembled $80.00
Cables (two needed per board) Each $25.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connectors or by a signal from the peripherals. The interrupt request lines for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard $20.00 Kit $60.00 Assembled $80.00
Cables (two needed per board) Each $25.00

ELEKTRA HD-5 Cabinet for dual 5-1/4" floppy drives with power supply, line cord, fuse, power switch, and power cables to drives 150.00
ELEKTRA HD-5W As above but with EMI filter, fan, and heavy duty power supply. Powers 1 floppy and 1 Winchester 199.00
5" ribbon cable for dual 5-1/4" disk drives 40.00
ELEKTRA D-8 Dual drive cabinet, power supply, ps cable for 8" drives 350.00
8" ribbon cable for dual 8" disk drives 45.00

GOLD 10 PIN CONNECTORS (Specify male w/square pins or female) 1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female) 0.50

Prices and inventory are subject to change without advance notice
This ad is our catalog

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

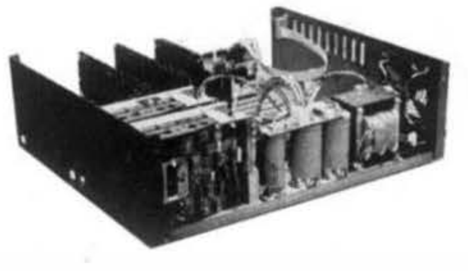
AAA CHICAGO COMPUTER CENTER
120 CHESTNUT LANE • WHEELING, IL 60090
(312) 459-0450
Technical consultation available 4 PM to 6 PM most weekdays. Closed evenings and weekends.

TERMS Minimum order $20.00. Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the mail. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

SS/DD DISK DRIVES	2 heads	1 head	2 heads
30 day guarantee	CDC	MPI	MPI
5 1/4" 40 tracks	300.00	225.00	275.00
5 1/4" 80 tracks	375.00	275.00	350.00
MPI or CDC Service Manual (Specify 40 or 80 track)			25.00
8" 77 tracks DS/DD Qume DT-8 $550.00			Remex (Special) 350.00

ELEKTRA
COMPUTER PRODUCTS



SMOOTH™ Software

(All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM

Single character commands for simplicity

Hardware independent (No interrupts required). Assumes modem is connected to an MC6850 (serial interface) and the control terminal is connected to an MC6850 (or MC6820 when used with the video version of GMXBUG™)

Transmit manually to distant computer

Transmit disk files (text) of any length to distant computer

Receive and save disk files (text) of any length on local disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested for full duplex operation at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo

Echo option so user can simulate a time sharing system. (Super Modem Program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces CR with CR/LF (user option) for those using time sharing systems that don't transmit LF's

Slow disk file transmit based on character verification (plus user installed timing loops if necessary) for use on time sharing systems to which disk files cannot be sent at speeds suggested by the baud rate

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8"
Manual and disk with both source and object code $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8"
Manual with instructions, source listing, and flow chart, disk with both source and object code $45.00
Manual with instructions, source listing, and flow chart $10.00
OS9 Modem Program Program $100.00

MODEMS (by U.S. Robotics)	MANUAL	AUTO	AUTO-ANS
Auto-Dial is Hayes compatible	ANSWER	ANSWER	AUTO-DIAL
300/1200 Baud, direct connect	N/A	$399/499.00	499.00
1200 Baud (120 cps), direct connect	349.00	399.00	N/A
300 Baud (30 cps), direct connect	169.00	199.00	N/A
300 Baud (30 cps), acoustic	129.00	N/A	N/A

THESE ADS ARE OUR CATALOG

ANOTHER WINNER! AAA Chicago Computer Center is proud to announce the addition of the **HELIX** line of computer products. The **HELIX** line, together with SWTPC, SSB and ELEKTRA permits us to offer you the greatest selection of quality components to handle all of your needs.

64K 6809 Computer	$2395 00	256K 68000 Computer	$3695 00
256K 6809 Computer	2895 00	512K 68000 Computer	4195 00

Other systems available

20 Megabyte 5" add on Winchester System			$2595 00
64K CMOS Static memory board with battery backup			395 00
DMA 5" and 8" Floppy Controller with built in Winchester controller I/O			695 00
DMA 5" and 8" Floppy Controller	$495 00	6809 CPU Board	495 00
68008 board for SS-50	495 00	68000 CPU Board	995 00

SMOOTH™ Software

ALL IN ONE
Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with cursor editing, character deletion and insertion), overstrike (for selected darker text), print, restart, set, top, underline, up and verify.

Supports Text Processing commands such as block copy, block move, centering, margin justification (widen and narrow), paging, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally addressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end of another, or insert (merge) one file into another as designated by the line pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. Control characters can be sent to the printer for format control either directly from the control terminal or by imbedding them in the text. The set command contains interface initialization and character output routines to support the SWTPC MP-C interface as well as the standard serial and parallel interfaces. Jumps are also provided to user supplied printer routines. User selects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer software routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (Warm Start) without destroying previously prepared text in the buffer. The Restart command erases contents in the buffer without the user having to reload the Editor.

The Editor allows the user to toggle between full duplex (no echo) and half duplex (echo) as needed. It responds to commands in both upper and lower case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 or 6809, SSB or FLEX™, 5" or 8"	$50 00
Printed source listing is available for an additional	35 00
All-In-One, Write'n Spell, and Spell'n Fix package	250 00

Software by Technical Systems Consultants, Inc.	UniFLEX™ w/1 yr mnt	FLEX™
DOS (includes Editor and Assembler)	550 00	150 00
Editor or Assembler		50 00
68000 Cross Assembler on 6809	300 00	250 00
6809 Cross Assembler on 6800 or 6800 FLEX™ Utilities		100 00
Text Processor or Sort/Merge Package or 6809 FLEX™ Utilities	150 00	75 00
Extended Basic	200 00	100 00
Basic Precompiler (specify standard or extended)	150 00	50 00
Pascal	300 00	200 00
Debug Package or Diagnostic Package	75 00	75 00
6809 Relocating Assembler & Linking Loader	175 00	150 00
Fortran (Requires Relocating Assembler & Linking Loader)		275 00
Fortran (With Relocating Assembler & Linking Loader)	450 00	375 00
Cobol	750 00	500 00

SMOOTH and **ELEKTRA** are trademarks of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.

6 NEW ELEKTRA PRODUCTS

ELEKTRA 30 PIN PROTOTYPING BOARD	20 00
ELEKTRA 50 PIN PROTOTYPING BOARD	40 00

ENGINEER'S "FUN BOX" by ELEKTRA Computer cabinet with high quality 10 amp power supply, EMI filter, Fan. Large enough to hold the standard size SS-50 type motherboard. 225 00

64K STATIC MEMORY BOARDS with gold connectors (kit available). Assembled and tested. With 56K RAM installed $299 00. With 64K RAM installed $349 00

12 and 24 MEGABYTE ELEKTRA ADD-ON WINCHESTER SYSTEMS (See below for pricing)

ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal print-out forms. Computer program can be used to write up purchase orders, packing list, account statements, plus more. Program is written in 6809 assembler language and does not require reverse video. Routines are included to support screen wrap around in both directions in case your video terminal does not. Source is provided so the customer can modify the program if he wishes. Comments are provided for adaptation to Lear Siegler and Televideo CRT terminals. The program can be easily adapted to other terminals if HOMEUP, HOMEUP and ERASE, and direct cursor addressing codes are known. CR, LF, and BS are the only other codes used. Available for 6809 FLEX. $100 00

SPECIALS

* Modems (Up to $100 00 off list). See our ad on the previous page			
* SSB BFD Floppy DiskControllers (Version 3) Run FLEX or SSB DOS			150 00
* MICROBUG (2K, 6809, Baby HUMBUG by Peter Stark)			30 00
* Cat 300 Baud Acoustic Modem (only 1 left)			75 00
* SWTpC 4K Memory $15 00, 8K Memory $40 00, MP-C $10 00, MP-LA			$80 00
		*SWTPC MP-2A	$125.00
* High speed tape reader	$50 00	Tandon SS 5" Drives	$150.00
* Remex 8" DS/DD (Special)	$350 00	* Ribbons for MX-80	$5.00

4K 6809 HUMBUG	75 00
4K 6800 HUMBUG (RAM needed at $A000 and $0-0001)	6 00
Other HUMBUG versions including video versions are available	
Spell'n Fix by Peter Stark	178 58
Write'n Spell by Peter Stark	75 11
Dynamite Disassembler	60 00
SUPER SLEUTH Disassembler System ($101 00 for OS-9 version)	99 00

Software by Microware Systems Corp.	Run-Time Package	Update	Source	Manual Only	Object w/Man.
OS-9™ Level One Operating System		75 00	400 00	40 00	200 00
OS-9™ Level Two Operating System		75 00	N/A	40 00	500 00
BASIC09™	100 00	75 00	N/A	25 00	200 00
OS-9™ Macro Text Editor			300 00	15 00	125 00
OS-9™ Interactive Assembler			300 00	10 00	125 00
OS-9™ Interactive Debugger (Disk version)			100 00	10 00	50 00
CIS Cobol Compiler	400 00	50 00	N/A	80 00	800 00
Pascal Compiler	100 00	100 00	N/A	40 00	400 00
Microware yearly support service ($200 00 for OS-9 Level 2)					75 00

SWTPC

6809 SWTPC FLEX™ Disk and manual (Disk only 15 00)			35 00
DC-4 Disk Controller (SS/DS, SD/DD, 5-1/4")			230 00
S32 Universal Static Memory Board (without memory chip)			124 50
SBUG-E (2716 compatible)	25 00	MP-R 2716 Eprom Programmer	114 50
MP-S2 Dual Port Serial	120 00	MP-L2 Dual Port Parallel	120 00
MP-N Calculator Board (kit)	54 95	MP-N (assembled)	92 00
MP-T Interrupt Timer (1 left)	92 00	MP-09 2MHz 6809 CPU Board	295 00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549 00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75 00
SE92/SA92-5 (6809 Edit/Assm for DOS)	69 95
SSB Monitor (Specify 6800/6809, $C000/$E000-$F7E0)	75 00
SSB version of FLEX™ (without Editor and Assembler)	150 00
LMB-1A Motherboard	399 00
SCB-69 6809 CPU Board	399 00
PAR-1 Dual Port Parallel Board	89 00
SER-2 Dual port Serial Board with 2 Cables	129 00
Chief 90 64K Computer System	2195 00

OS-9 and **BASIC09** are trademarks of Microware Systems Corp.
GIMIX and **GHOST** are registered trademarks of GIMIX Inc.

THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! — THE ELEKTRA SUPER FC

Controls up to four 5 1/4" drives and four 8" drives for a total of eight system drives for both 6800 and 6809 systems. (FLEX software has a built in limit of a system mix of four drives). Improved programmed input/output (PIO) interface that allows single and double density 5 inch operation at 1MHz in both 6800 and 6809 systems. Allows single density 8 inch operation at 1MHz in both 6800 and 6809 systems. Allows both single and double density operation of both 5 inch and 8 inch drives at 2MHz in both 6800 and 6809 systems without slow device circuitry. Hardware and software compatible with existing SWTPC DC-1, DC-2, AND DC-3 type controllers. Controls virtually any floppy disk drive, any mix of drive size, recording density, track density, number of sides, motor control, with or without head load solenoids, including the new 2 Mbyte 5 inch drives. Analog phase locked loop data separators for maximum reliability. Provides separate adjustments for 5 inch and 8 inch drives. Analog write precompensation circuit with separate adjustments for 5 inch and 8 inch drives. Designed to meet the data hold requirements of Western Digital floppy disk controller IC. Schmidt trigger hysteresis input buffers for disk drive signals. Software write protect control. Interrupt/Data Request enable control. 12 option jumper areas to select just the right combination of hardware features to match your drives and software. Provides many separate option selections for 5 inch and 8 inch drive operation. Gold bus and drive cable connectors. Test connector for easy setup and field service. Single 5 volt power supply. 4 or 16 addresses per slot decoding. Ideal replacement for existing controllers to add double density data storage and 8 inch drive capability. Disk driver software and disk formatting utilities available to extend the capabilities of your current software. Get more for less with the ELEKTRA Super Floppy Controller.

ELEKTRA Super FC Super Floppy Controller (assembled and tested)	$275.00
Disk with drivers and formatting utilities (Specify 6800 or 6809)	$30.00
TSC General FLEX or FLEX 9 both with Editor and Assembler	$150.00

THE BEST WINCHESTER DISK SYSTEM THAT YOU CAN BUY! — THE ELEKTRA WINCHESTER

Has automatic error detection and **CORRECTION** of up to 11 bit burst errors.

SS-50 bus, extended addressing capabilities, DMA, on board sector buffer, operates with 6800 and 6809 type processors, drivers included for FLEX 9 or OS-9. Drivers for FLEX 2 (6800) available for an additional $100 00. Price includes host interface, controller, drivers, and cables.

12 Megabyte ELEKTRA single drive Winchester system	$2295.00
24 Megabyte ELEKTRA dual drive Winchester system	$3595.00
19 Megabyte ELEKTRA single drive Winchester system	$2995.00
38 Megabyte ELEKTRA dual drive Winchester system	$4695.00
ELEKTRA heavy duty regulator bd. (powers 1 floppy and 1 Winchester)	$75.00
ELEKTRA SHD-5 cabinet, ps, ps cables (powers 1 floppy and 1 Winchester)	$199.00

AAA Chicago Computer Center See our ad on the previous page to your left for ordering instructions.

Technical Consultation available 4 p.m. to 6 p.m. most weekdays. Closed evenings and weekends.

WINDRUSH
Micro Systems Ltd.

[illegible]
Tel 0692 405189
Telex 975801 [illegible]

'68' MICRO JOURNAL
New Products Editor
P.O. Box 849
Hixson
Tennessee
37343 U.S.A.

Your Ref — Our Ref WCD/sg 356 Date 6/8/83

THE LOW-COST 3U EUROCARDS FORM HEART OF TERMINAL OR 6809 DEVELOPMENT SYSTEM

Windrush Micro Systems Limited announce the immediate availability of two 3U eurocards which can be used to build a standalone terminal, an MC6809 development system, or stand alone industrial/process controllers. The products feature on-board regulators which eliminate the need for regulated D.C. supplies and a low profile which enable the boards to be installed on a 3E pitch.

PROCESSOR BOARD

- 2.0 MHz MC68B09 CPU is fully buffered from the bus.
- 16K of EPROM and 2K of static RAM may be link-configured to form a 6809 development system running the TSC 'FLEX' disk operating system.
- Serial port via a programmable R6551A ACIA w/crystal controlled baud rate generator is fully buffered to RS-232 levels to a standard DB-25 connector.
- Twin parallel port, two 16-bit timers and serial interface provided by a 6522A VIA which is interfaced via a standard 26 way IDC header.
- Includes 'WT-BUG9' system monitor which provides a comprehensive range of system monitor functions and includes all software necessary to implement an intelligent terminal with the Windrush 3U video board. Assembly language source is supplied.

VIDEO BOARD

- Programmable MC68B45 CRT controller provides a variety of display formats from 80 x 24 (standard) to 32 x 16.
- 2K of memory mapped video RAM may be mapped to any 1K boundary in the system.
- A user programmable character generator is supplied in a 4K EPROM. The standard character set provides a full range of ASCII characters, including control codes and lower-case with true descenders, in an 8 x 12 cell. A reverse video attribute is also provided. Software to assist the user in developing custom character cells is also available in EPROM or on disk.

Other 3U boards include: serial/parallel I/O boards, A/D and D/A boards, memory boards, and a double density floppy disk controller which has drivers available for the 'FLEX' disk operating system.

CMS CREATIVE MICRO SYSTEMS
3822 CERRITOS AVENUE LOS ALAMITOS CALIFORNIA 90720 U.S.A. (213) 493-2484

M6809 MICROCOMPUTER SYSTEM

THE 9687 IS A COMPLETE MICROCOMPUTER IN A SINGLE STANDALONE ENCLOSURE. IT IS AVAILABLE IN A VARIETY OF CONFIGURATIONS WITH RESPECT TO BUILT-IN WINCHESTER AND FLOPPY DISC STORAGE FEATURES, MEMORY CONFIGURATIONS AND INPUT/OUTPUT OPTIONS. THE 9687 IS A TABLETOP MODEL WITH 14 CARD POSITIONS INTERNALLY. IT FEATURES FORMATTED DISC CAPACITIES UP TO 50 MEGABYTES FOR THE WINCHESTER DRIVE AND ONE MEGABYTE FOR THE MINI DISKETTE DRIVE. IT CAN ACCOMMODATE UP TO ONE MEGABYTE OF RANDOM ACCESS MEMORY. THE 9687 CAN PROVIDE UP TO TEN SERIAL PORTS AND A VARIETY OF OTHER I/O ARRANGEMENTS.

THIS MICROCOMPUTER CAN BE EQUIPPED WITH OS-9, A SOPHISTICATED MULTIUSER MULTITASKING OPERATING SYSTEM. THIS COMBINATION OF HARDWARE AND SOFTWARE PROVIDES AN IDEAL SOLUTION FOR MANY INDUSTRIAL AND BUSINESS APPLICATIONS.

THE SINGLE QUANTITY PRICE FOR THE 9687 WITH MINIMUM OPTIONS IS $5995.00. QUANTITY DISCOUNTS ARE AVAILABLE. DELIVERY IS FROM FACTORY STOCK.

ADDITIONAL INFORMATION IS AVAILABLE FROM CREATIVE MICRO SYSTEMS, 3822 CERRITOS AVENUE, LOS ALAMITOS, CA 90720. TELEPHONE (213) 493-2484.

LLOYD I/O
19535 NE GLISAN
PORTLAND, OR 97230

FRANK L. HOFFMAN
(503) 666-1097

6809 COMPUTER SOFTWARE: EDITORS, ASSEMBLERS, COMPILERS

NEW PRODUCT

LLOYD I/O introduces the OS9 JOB CONTROL LANGUAGE called "DO". This new language is 4K in length and is intended to be used for batch processing. Its appearance is similar to BASIC. DO reads the procedure file from disk in its entirety. The price for the 5.25" or 8" disk and 30 page manual is $[illegible].00

DO uses a total of about 8.5K of user memory minimum but may be assigned more memory when the user's application procedure is larger than the buffer. DO has 26 number variables and nine string variables. Passing parameters to DO for substitution is allowed and in fact is only limited in number to the size of the OS9 command line. There are 17 directives of which most are just like BASIC statements. DO allows labels for controlling the flow of execution. When an error is detected the ON ERROR GOTO traps it and allows the user's procedure to take the correct action.

DO procedures are easy to write. Labels can be any length and any series of characters with no imbedded spaces. Comments are allowed but are not loaded into memory when DO reads a procedure from disk. This saves memory space. The directives include: REMARK, TRACEON, TRACEOFF, LET, GOTO, GOSUB, RETURN, ON ERROR GOTO, RESUME, REPORT, IF, IFC, PRINT, INPUT, READ, END, CHAIN. Fatal errors (syntax errors) result in an error message with the line and a pointer to the character where the error was found.

DO procedures can be chained or nested by using the DO command as an OS9 command. The manual is thirty pages long and includes a procedure library of examples. The manual also includes a directive summary appendix for handy reference.

OS9 is a trademark of Microware Systems Corporation

SOFTWARE DEVELOPMENT TOOLS from LLOYD I/O

E D

A Programmer's Editor for FLEX and OS9

ED is a combination "Screen" and "Line" editor with extended features that allows the programmer to write programs for any computer language or even word processing files such as letters and manuscripts. ED operates identically on FLEX and OS9.

FEATURES:

- edits files larger than memory
- extended disk file functions
- restart editing with out reloading
- automatic start up procedure file
- redefineable keyboard
- works with any terminal
- calculator for integer expressions
- command macros with parameters

O S M

6809 MACRO ASSEMBLER for FLEX and OS9

OSM is a MACRO assembler with CONDITIONAL assembly directives and other extended commands that are not found in other assemblers. If you write programs for OS9 and/or FLEX then OSM is for you because it generates OS9 or FLEX machine language binary files under OS9 or FLEX.

FEATURES:

- Motorola 6809 standard mnemonics
- library file calls nestable
- conditional assembly nestable
- macros nestable, with parameters
- up to 30 character symbols
- local labels (nestable)
- automatically generated labels
- errors tell file name and line
- object code format for OS9 or FLEX

ORDER FROM: LLOYD I/O
19535 NE GLISAN STREET
PORTLAND, OR 97230
TELEPHONE: (503)-666-1097

Visa or Master Charge Accepted

C R A S M B

MACRO CROSS ASSEMBLER for FLEX and OS9

CRASMB is a MULTI-CPU Cross Assembler. This is our best assembler. CRASMB is a macro-conditional assembler. It uses machine language overlays or modules called "CPU Personality Modules" to do the work of mnemonic look up. It has directives and other extended commands that are not found in other assemblers. It generates OS9 or FLEX binary files.

FEATURES:

- Cross assembles 8 CPU types: Motorola 6800-2-8, 6801-3, 6805, 6809 Mostek 6502, RCA 1802, INTEL 8080-5 ZILOG Z-80
- Manufacturer's standard mnemonics
- library file calls nestable
- conditional assembly nestable
- macros nestable, with parameters
- up to 30 character symbols
- local labels (nestable)
- automatically generated labels
- errors tell file name and line
- object code format for OS9 or FLEX

Other Products from LLOYD I/O:

ASM (FLEX 6809 Macro Assembler)
ISM (FLEX 6809 Interactive Assembler)

* P R I C I N G *

name	FLEX	OS9
ED (editor)	$50	$50
CRASMB (cross assembler)	$200	$200
each CPU module binary	$35	$35
with source code	$70	$70
All eight w/source	$498	$498
OSM (OS9/FLEX assembler)	$99	$99
ASM (FLEX assembler)	$50	n/a
ISM (interactive asm.) free with FMS Flex		

(Flex is a trademark of TSC)
(OS9 is a trademark of Microware)

OEM Licenses Available

GENTLEMEN:

PLEASE SEND YOUR FLEX USERS NOTES WITH 8 INCH DISK.

SINCE FROM THE ADDRESSES I SEE IN THE MICRO JOURNAL, YOU ARE PART OF OR RELATED TO THE 68 MICRO JOURNAL, I'LL SAY TO EVERYONE AT THE SAME TIME....GOOD JOB!

MICRO JOURNAL AND THEIR LONG TERM ADVERTIZERS HAVE KEPT THE 68XX ALIVE. I THANK EVERYONE

FOR THAT. KEEP UP THE GOOD (AND THANKLESS) WORK.

SINCERELY,

Ken

KEN WILSON
6294 CLOVER COURT
CHINO, CALIFORNIA
91710

P.S. WHERE THE HELL IS HIXSON, TENN?

Ed's Note: Golly Ken I thought everyone knew. Hixson is about 10 miles South of Soddy-Daisy, at the foot of Big Ridge. By the way, where in the heck is California?

DMW

- - -

Southeast Media Supply
Data Comp. Division - CPI
5900 Cassandra Smith Road
Hixson, TN. 37343

Reference: Invoice 4427

I want to thank you for your prompt service in obtaining an update of my old Lucidata Pascal. It arrived the week before last and was immediately put into service. As you can conclude from the attached copy of my letter to Lucidata, I am quite pleased with its ability to handle heavy duty programming.

Since I received my update directly from Lucidata, I was wondering if you knew that this task was completed. If there are any additional charges for this service, please notify me.

By the way, if you can help me with my FLEX-09 to CP/M disk translation problem, I would appreciate your letting me know.

Sincerely,

Philip C. Nunn

Philip C. Nunn
201 Netherfield
Comstock Park, MI. 49321

Lucidate Ltd.
P.O. Box 128
Cambridge CB2 5EZ
England

I want to thank you for your prompt response in updating my Pascal system. I did not realize just how outdated it was until I tried to install the INCLUDE utility. The other improvements are quite helpful to me. I am working on a large project to update a research management information system which eventually will be run on a PDP-11/70. Lucidata and the Decus Pascals are both quite faithful to the ISO standard, so my source code recompiles easily on the PDP-11/70. These programs are relatively large, 15K minimum compiled, and generously modularized. Consequently, I have been using your Pascal heavily for several weeks, and am impressed with the high quality you have provided.

The only problem I have encountered is with a small program which accesses an external procedure to read the system clock in my Gimix 6808+. The program accesses the procedure once each second. It works fine for about 8 cycles then dies. It seems as if some counter in the run time system is overflowing. I would appreciate any advice you can give me to change this condition. I have included a listing of the source program for reference. Incidently, you are welcome to use this program as you wish. It is the least I can do in return for any assistance you might be able to offer.

I was pleased to get your updates of the Copycat disk utilities. I was especially pleased with the CP/M 8-inch ones. I had modified your old 5.25-inch C /M disk utilities to work with my 8-inch CP/M disks, but I like your new ones better than mine. My Gimix 6809+ has a Metalab 2809 board in it. It works very reliably and smoothly runs such CP/M software as Wordstar (used for this letter) and dBASE II. However, all of my system control and utilities are still in 6809 code and manageable only with FLEX-09. Your Copycat utilities have provided me with a much needed link between the two operating systems. Being able to copy disk files from my CP/M disks to my FLEX-09 disks will now permit me to use the full power of my system on them for such tasks as telecommunications, double density double sided disk storage, and temporary storage in my 112K of secondary memory (with VDISK). Unfortunately, you have provided me with only half of the transfer capability which I need. I desperately need a FLEX-09 utility which can write a file on a CP/M standard single density single sided 8-inch disk. Considering the inflexible disk format used by the CP/M 8-inch standard, it seems like it should be a realtively straight forward coding task, but I do not know of such a program. If you have one even in draft form, I am interested in purchasing it. Or if not, can I interest you in writing it?

I have enjoyed my association with you, and look forward to future opportunities. I hope that you can be of assistance to me in trying to solve my two problems.

Sincerely,

Philip C. Nunn

Philip C. Nunn
201 Netherfield
Comstock Park, Michigan 49321
U.S.A.

ACORN™

COMPUTER SYSTEMS SS-50C

Calculate

```
****************************************************
*  CALCULATOR  DEC,HEX,BIN--ADD,SUB,CONVERT        *
*    uses FLEX (TM) subroutines                    *
*                                                  *
*              By Merle Giesfeldt                  *
*        ACORN COMPUTER SYSTEMS, Inc.              *
*                                                  *
* SAMPLE :                                         *
*  Calc(DHB+-=) HF5E3 - B0001 1010 + D678 = H      *
*  H=xxxx                                          *
****************************************************

        ORG     $100

CALC    LDX     #MATHST    print heading
        JSR     $CD1E      PSTRNG FLEX
        JSR     $CD1B      INBUFF FLEX
        JSR     GETNUM     get first number
        PSHS    X          store it in buffer

DDMATH  CMPA    #'+        (plus)
        BEQ     ADD
        CMPA    #'-        (minus)
        BEQ     SUB
        CMPA    #'=        (equal)
        BEQ     EQUAL
        BRA     QUIT       entry error

ADD     JSR     GETNUM
        PSHS    A          save sign
        TFR     X,D        put number in D reg
        ADDD    1,S        add numbers
        BRA     SUB2

SUB     JSR     GETNUM
        PSHS    A          save sign
        LDD     1,S        put buffer in D reg
        STX     1,S        put new num in buffer
        SUBD    1,S        subtract numbers
SUB2    STD     1,S        save answer in buffer
        PULS    A          get math sign
        BRA     DDMATH
EQUAL   LDA     #$0A       (line feed)
        JSR     $CD18      PUTCH
EQU1    JSR     $CD27      NXTCHR FLEX
        CMPA    #$20       (space)
        BEQ     EQU1       skip over spaces
        JSR     $CD1B      PUTCH FLEX
        PSHS    A          temp save base code
        LDA     #'=
        JSR     $CD1B      PUTCH
        PULS    A          get base code
        CMPA    #'H        hex number ?
        BEQ     OTHEX
        CMPA    #'B        binary number ?
        BEQ     OTBIN

* OUTPUT decimal if no base code
        LEAX    ,S         load addr of answer
        CLRB               suppress leading zeros
        JSR     $CD39      OUTDEC FLEX
        BRA     QUIT       done
```

```
OTHEX   LEAX   ,S          load addr of answer
        JSR    $CD45       OUTADR FLEX
        BRA    QUIT        done

OTBIN   LEAX   ,S          load addr of answer
        BSR    OBINI       output binary number
QUIT    LEAS   2,S         restore stack
        JMP    CALC        next calculation

OBIN1   BSR    OBIN2       output first byte
OBIN2   LDA    ,X+         get byte
        BSR    OBIN3       output nibble
OBIN3   LDB    #4          bit count
OBIN4   LSLA               test bit
        PSHS   A           temp save byte
        BCC    OBIN5       if bit = 0
        LDA    #'1
        BRA    OBIN6
OBIN5   LDA    #'0
OBIN6   JSR    $CD18       PUTCHR FLEX
        PULS   A           restore byte
        DECB               dec bit count
        BNE    OBIN4       if nibble not done
        PSHS   A           save byte
        LDA    #$20        (space)
        JSR    $CD18       PUTCH
        PULS   A           restore byte
        RTS

GETNUM  JSR    $CD27       NXTCH FLEX
        CMPA   #'X         quit ?
        LBEQ   $CD03       WARMST
        CMPA   #'D         decimal number ?
        BEQ    DEC
        CMPA   #'H         hex number ?
        BEQ    HEX
        CMPA   #'B         binary number ?
        BEQ    BIN
        CMPA   #$20        (space)
        BEQ    GETNUM      skip spaces
        BRA    QUIT        entry error

BIN     LDX    #$0000      clear buffer
BIN2    JSR    $CD27       NXTCH FLEX
        CMPA   #$20        (space)
        BEQ    BIN2        skip spaces
        CMPA   #'0         (zero)
        BNE    BIN3
        TFR    X,D         transfer buffer to D
        LSLB               shift in bit
        ROLA
        TFR    D,X         save in buffer
        BRA    BIN2        get next bit
BIN3    CMPA   #'1         (one)
        BNE    HEX3        if all bits entered
        TFR    X,D         transfer buffer to D reg
        LSLB               shift in bit
        ROLA
        ADDD   #1          make bit = 1
        TFR    D,X         save D in buffer
        BRA    BIN2        get next bit

DEC     JSR    $CD48       INDEC FLEX
        BRA    HEX2

HEX     JSR    $CD42       GETHEX FLEX
HEX2    JSR    $CD27       NXTCH get sign
        CMPA   #$20        (space)
        BEQ    HEX2        skip spaces
HEX3    RTS

MATHST  FCB    $0A,$0D,$0A
        FCC    "      (X=exit)",$A,$D
        FCC    " Calc(DHB+-=) ",4

        END    CALC
```

ACS INC

11931 W. BLUEMOUND ROAD
WAUWATOSA WISCONSIN 53226

1129 Lois Drive
Cincinnati, Ohio 45237
20 August 1983

Computer Publishing Center
68' Micro Journal
5900 Cassandra Smith
PO Box 849
Hixson, Tennessee 37343

Dear Don,

Here's one for the bit bucket. I discovered it while writing my 6809 disassembler.

NEGA is defined as COMA + 1. The carry bit is adjusted. However, the carry bit is the compliment of what you would expect. (The compliment of 00 is FF. FF + 1 = 00 with a carry. But the carry is 0!)

There is a subtle reason for this, namely it makes things easier when dealing with multiple byte numbers. For example, using a five byte number starting at $TNEG, with the carry set as expected (FF + 1 = 00 and a carry), the code would look like this:

```
        ...
        LDB    4         *NUMBER OF BYTES - 1
        LDX    #TNEG     *FIRST BYTE
LOOP1   COM    0,X+       *COMPLIMENT ALL BUT LAST
        DECB
        BNE    LOOP1
        LDB    4         *BYTES - 1
        NEG    0,-X       *NEGATE LSB
LOOP2   LDA    0,-X       *NOW RIPPLE THE CARRY
        ADCA   0
        STA    1,X
        DECB
        BNE    LOOP2
        ...
```

However, with the carry inverted, the code looks like this:

```
        ...
        LDB    5         *NUMBER OF BYTES
        LDX    #TNEG+5   *LAST BYTE + 1
        ANDCC  $FE       *CLEAR THE CARRY
LOOP    LDA    0,-X       *GET NEXT BYTE
        ADC    0         *ADD THE CARRY
        NEGA              *NEGATE IT
        STA    0,X        *STORE IT
        DECB

        BNE    LOOP       *NOT DONE
        ...
```

Not only is the code shorter, and therefore faster, it is also more clear. There is no special handling of the last byte.

Although this originally threw me, it demonstrates an old truth. THE TROUBLE IS NOT IN YOUR SET. PLEASE ADJUST YOUR HEAD!

Sincerely,

William R Stock

William R. Stock

Mr. E. Fyfe,
2 Hellebore Court,
Mildmay Road,
Stevenage,
Herts., SG1 5TA,
England.
9th August,1983.

Dear Don,

Let me first say thank you to all at 68' Micro for producing such a great magazine. I have subscribed to it since late 1979, but this is the first time that I written to you, apart from ordering my subscriptions. I have recently installed a daisywheel printer - until that time, I did not have any printing facilities.

I have a 6800/6809 system running with two 5 inch floppy disk drives. I have installed both the 6800 and the 6809 General Flex on my system, which uses the Gimix DMA controller board.

The item which triggered this letter was the letter from Bud Pass in the August '83 issue (page 37). In this letter, he states that there is a serious bug in the TSC Basic, and quotes an example program line which demonstrates the 'bug'.

In the TSC XBasic Manual section on the 'IF-THEN-ELSE' statement, the syntax specification shows only one statement between the 'THEN' and the 'ELSE'. His example has two statements in this position. The basic interpreter will regard the line terminator for the first 'PRINT' statement as the terminator for an IF-THEN statement, and will thus ignore the remainder of the line, if the condition is false.

The point to note is that the 'IF-THEN-ELSE' statement is ONE statement, and thus any terminator found will terminate that statement. The only way that the line quoted by Bud could be made to work, would be if a 'BEGIN...END', or similar construct, is available in the Basic used.
On a different topic, on page 35 of the March '83 issue, Don Aldridge gave the 'ultimate in memory clear programs', comprising five instructions. It is however possible to do the same thing with two instructions, namely:

```
                  VALUE EQU  $0000
                        ORG  VALUE-6
FFFA    10CE FFFE START LDS  #START+4
FFFE    8D   FE   CLEAR BSR  CLEAR
                        END  START
```

VALUE represents the value to which the memory is to be initialised by the routine. The routine works by setting up the stack pointer such that the first return address stacked, will overwrite the immediate value which was loaded into the stack pointer. This instruction is followed by a subroutine call, which calls itself, resulting in an 'infinite' loop. Subsequent subroutine calls will continue decrementing the stack pointer until, eventually, the last call will overwrite the BSR instruction. I have not been able to try this routine as shown, because I have Prom at these locations, however, I have tested the routine assembled at other origins.

The two instruction sequence can be used to initialise a 6809's memory to any value (or pair of values!) by reassembling the routine with a new VALUE. A similar routine can be generated for the 6800.

I hope that the above discussions have provided some useful information.

Sincerely,

Eric Fyfe

Eric Fyfe.

John H. Deal
122 Moorings Park Drive Apt. 709
Naples Florida 33942
1-813-261-0536

BASLIST

August 5,1983
Page No. 1

'68' MICRO JOURNAL
5900 Cassandra Smith - P.O.Box 849
Hixson Tennessee 37343

Dear Don:

Yesterday I mailed you a listing of a small program called 'BASLIST', dated August 04 1983. I have just discovered that the program I sent was not the final version. Please destroy that one DATED AUGUST 04 1983 and substitute the enclosed correct one DATED AUGUST 05 1983.

The program was written for use with an OKIDATA 83A printer and the appearance of CHR$(29) and CHR$(30) are printer designators for 16.5 and 10 C.P.I. respectively. If the user's 16.5 c.p.i. is not available the subroutine for 16.5 c.p.i. can be eliminated, as can also any reference to CHR$(30).

The author of the original program has long since been lost in my memory and credit to him is now extended. I have enhanced the header with information for filing and identification and provided for left indent of any wrap around portions of long lines. The left margin may be set by changing the value of X in lines 370 & 470.

The wrap around does not split words and I think the end result is a more pleasant looking listing of a basic program. Perhaps you will find this program suitable for inclusion in your Journal at some date. I hope others may want to add it to their utility file.

Very truly yours,

John H. Deal

John H. Deal

CC:Disk

```
100 REM .. THIS PROGRAM CALLED ------------------  ' B A S L I S T '
110 PRINT CHR$(12)
120 PRINT TAB(25);"BASLIST"
130 PRINT TAB(20);"FOR FLEX 9.0 XBASIC":PRINT:PRINT
140 PRINT"THIS PROGRAM WILL LIST A DISK BASIC FILE AND ALLOWS"
150 PRINT"YOU TO SET THE LEFT MARGIN AND THE HEADER"
160 EXEC,"TTYSET DP=0,WD=0"
170 INPUT"WHAT IS THE BASIC PROGRAM TO BE LISTED (AS 'BASLIST')";B$
180 B$=B$+".BAS":X=10:C=0
190 INPUT"WHAT IS THE DISK NUMBER (AS 106)";F
200 INPUT"DO YOU WANT LISTING AT 16.5 CPI (Y/N)";K$
210 INPUT"WHAT IS TODAYS DATE (AS JUNE 12 1983)";D$
220 OPEN "0.PRINT" AS 0
230 GOSUB 360
240 IF LEFT$(K$,1)="Y" THEN GOTO 450
250 ON ERROR GOTO 340
260 OPEN OLD B$ AS 1
270 INPUT LINE #1,L$
280 IF LEN(L$)>(80-X) THEN GOSUB 590:GOTO 270
290 PRINT #0,TAB(X);L$
300 IF C=53 THEN PRINT #0,CHR$(12);GOSUB 360:C=0:GOTO 270
310 C=C+1
320 GOTO 270
330 END
340 IF ERR<>8 THEN ON ERROR GOTO 0
350 RESUME 330
360 REM .. SUBROUTINE TO PRINT HEADING
370 C=0:P=P+1:X=5
380 P$="PAGE NO. "+STR$(P)
390 PRINT #0,CHR$(30)
400 PRINT #0,TAB(X);D$;
410 PRINT #0,TAB(25-X);"FLEX 9.0 - DISK NO. ";F;TAB(60+X);B$
420 PRINT #0,TAB(X);"JOHN H. DEAL";TAB(60+X);P$
430 REM
440 PRINT #0:PRINT #0:RETURN
450 REM .. SUBROUTINE FOR 16.5 CPI
460 ON ERROR GOTO 570
470 PRINT #0,CHR$(29):X=25
480 OPEN OLD B$ AS 1
490 INPUT LINE #1,L$
500 REM
510 PRINT #0,TAB(X);L$
520 IF C=51 THEN PRINT #0,CHR$(12);GOSUB 360:PRINT #0,CHR$(29)
530 C=C+1
540 X=25:GOTO 490
550 PRINT #0,CHR$(30);CLOSE 1:CLOSE 0
560 END
570 IF ERR<>8 THEN ON ERROR GOTO 0
580 RESUME 550
590 REM .. SUBROUTINE TO INDENT THE WRAP AROUND LINE
600 FOR J=1 TO 20
610 L1$=LEFT$(L$,(80-J))
620 IF RIGHT$(L1$,1)<>" " THEN 630 ELSE 640
630 NEXT J

640 L2$=RIGHT$(L$,(LEN(L$)-LEN(L1$)))
650 L1=VAL(LEFT$(L1$,4)):L1=LEN(STR$(L1))
660 PRINT #0,TAB(X);L1$
670 C=C+1
680 PRINT #0,TAB(L1+(X+1));L2$
690 RETURN
```

A PARTIAL LISTING SHOWING HEADER AND WRAP AROUND INDENTATION

```
AUGUST 05 1983            FLEX 9.0 - DISK NO. 1020            CKRECON.BAS
JOHN H. DEAL                                                  PAGE NO. 1

100 REM .. THIS PROGRAM CALLED .................. ' C K R E C O N '
110 PRINT CHR$(12)
120 I=400:J=1:P=1:DIM A$(I),B(I)
130 PRINT"THIS IS THE SUBROUTINE CALLED ..........  'C K R E C O N'"
140 PRINT"AND RECONCILES THE CHECK BOOK BALANCE WITH THE BANK
    STATEMENT":PRINT
150 OPEN "0.PRINT" AS 0
160 EXEC,"TTYSET WD=0,DP=0"
170 PRINT #0,CHR$(27);CHR$(66)
180 INPUT"HOW MANY ENTRIES ON THE BANK STATEMENT";N:K=N:PRINT
190 INPUT"WHAT IS THE LAST STATEMENT BALANCE";C$:PRINT
200 INPUT"WHAT IS THE LAST STATEMENT BALANCE DATE";D$:PRINT
210 INPUT"WHAT IS THE LOWEST OUTSTANDING CHECK NUMBER (IF NONE THEN ENTER
    'NONE')";T$
220 INPUT"WHAT PERIOD DOES THE STATEMENT COVER (AS 04-23-83 THRU
    05-31-83)";S$:PRINT
230 ON ERROR GOTO 1430
240 GOSUB 1400
250 INPUT"ENTER STATEMENT CHECKS ONE AT A TIME OR 'DONE' IF FINISHED";A$
260 REM
270 FOR R=1 TO I
```

Mr Don Williams Sr
Computer Publishing Center
68 Micro Journal
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
USA

PARAMOD..OS9

Dear Don,

Your magazine is excellent in enabling me to get more out of my 6809 computer. I am happy that you are incorporating more articles on OS9 in 68 Micro Journal. Peter Dibble is doing a very good job in explaining OS9.

I hope that you soon will get a sufficient number of European subscribers to allow for 68 Micro Journal being bulk delivered to Europe by jet like Electronics magazine. Now the air mail costs more than your magazine itself.

Here follows a description of a program that could be of interest to your readers, and I will be happy if you publish it.

PARAMETER LIST MODIFIER FOR BASIC09
****

Microware's BASIC09 is an excellent interpreter, easy to use for producing your own utilities. Unfortunately, it is lacking a straightforward method of passing parameters. For example, if you are going to write a "Help" utility, you want to type

    OS9:help dir

to learn about the dir command. This is impossible if Help is a BASIC09 program.

If Help is a packed BASIC09 program, interpreted by RunB, you can type

    OS9:help

only, and let the program ask you what help you want. If you don't have RunB, you have to type

    OS9:basic09 #5k help

However, even if Microware don't tell you, you can also pass parameters in RunB or BASIC09 by using the syntax

    OS9:help ("dir")   or   OS9:basic09 #5k help("dir")

by issuing a PARAM statement in the Help program. This is OK if you will use the program rarely. But if the program will be used often, and perhaps not by yourself, this is a very clumsy syntax.

The desired syntax can of course be accomplished by writing Help in another language that permits the desired parameter syntax, i.e. in assembler. This is probably the wrong way for a user utility program.

To solve the problem, I have written a short "universal" program in assembler, called ParamMod, with the following characteristics:

- ParamMod allows the desired parameter syntax.
- ParamMod transforms the parameter list from the desired syntax to the syntax required by BASIC09 or RunB. The resulting parameters are all of the type STRING. To be used as numeric types, the strings has to be transformed using the VAL function.
- ParamMod forks to either BASIC09 or RunB, and the main program is written in BASIC09.
- ParamMod has to be duplicated and customized on three text strings and needed BASIC09 memory for each utility:
  - inname: The wanted utility name. In the given case, Help. Other utilities could be named Compare or Analyze.
  - outname: The name of the file that contains the BASIC09 procedure and performs the desired action. It could be named Help_B or /D0/CMD/Compare_B or AnalyzeBody.
  - Interprt: The name of the BASIC09 interpreter to be forked to. Either BASIC09 if outname is a saved procedure or RunB if it is a packed procedure.
  - memory: The total number of bytes needed for the procedures and their data areas.

In the following, we are assuming you are writing a Help utility. For other utilities, change the names accordingly.

First customize ParamMod's three text strings and BASIC09 estimated memory size with your text editor. Then assemble it with Microware's assembler, using the command:

```
OS9:asm ParamMod o=help #10k
```

and the resulting code for Help will be in your execution directory.

Then write your BASIC09 program, naming the outermost procedure Help_B. You must save or pack Help_B to run it through Help. Run it from within BASIC09 with the command (including parameter):

```
B:$help dir      (note: $=dollar sign)
```

You may also during the development phase run the program without Help. In that case you must use BASIC09 parameter syntax:

```
B:run help_b("dir")
```

Included is the assembly listing for ParamMod, customized for a Help utility and a dummy Help_B BASIC09 program.

Sincerely,

Bengt-Allan Bergvall
Blavingev. 1
S-561 49 Huskvarna
Sweden

```
00001          * Program written by Bengt-Allan Bergvall, Blavingev. 1,
00002          * S-561 49 Huskvarna, Sweden.
00003          *
00004          * Program to reformat a parameter list from an easy typed form
00005          * to the clumsier form required running a BASIC09 program.
00006          * Given the command
00007          * OS9:help param1 param2  param4          (note the extra space)
00008          * this program will fork to the RunB or BASIC09 program
00009          * Help_B as if given the equivalent command:
00010          * OS9:BASIC09 #5k Help_B("param1","param2","","param4")
00011          *
00012          *
00013          * This program is general and can reformat the resulting parameter
00014          * list up to 256 characters, but the name strings inname and
00015          * outname has to be changed for each implementation.
00016          *
00017          * If interpt is RunB, then outname has to be a packed BASIC09
00018          * program in execution directory.
00019          *
00020          * If interpt is BASIC09 then outname has to be an saved BASIC09
00021          * program either in present data directory, or in an other file
00022          * with outname given the full path name, i.e. /d0/COM/Help_B
00023          *
00024          * The memory needed by BASIC09 or RunB must also be given.
00025
00026                              opt    f
00027
00028                              nam    parameter list modifier
00029                              ttl    for BASIC09 or RunB
00030    0000 87CD0092             mod    pgend,inname,prgrm+objct,reent+1
00031    0009 001F01CA             fdb    pgstart,stack
00032                              ifp1
00034                              endc               (use os9defs)
00035
00036          * data variables
00037 D 0000                       org    0
00038 D 0000           parend5     rmb    2          output parameter limit -5
00039 D 0002           outpar      rmb    256
00040 D 0102           varend      equ    .
00041 D 0102           stck        rmb    200        stack area
00042 D 01CA           stack       equ    .          stack pointer
00043          ***********************************
00044          ** Customization area
00045
00046    000D 48656CF0   inname   fcs    .Help.     Name of utility
00047    0011 48656C70   outname  fcs    .Help_B.   Name of BASIC09 procedure
00048    0017 00                  fcb    0
00049    0018 42415349   interpt  fcs    .BASIC09.  Either BASIC09 or RunB
00050          * Total memory needed in bytes by BASIC09 or RunB process:
00051          * (Equivalent to the needed BASIC09 MEM value)
00052    1388            memory   equ    5000
00053
00054          ** End customization
00055          ***********************************
00056
00057    001F            pgstart
00058
00059          * Modify parameter list from free form into BASIC09 string form.
00060          * Example of free form: param1 param2  param4
00061          * Resulting BASIC09 string form: Help_B("param1","param2","","param
Microware OS-9 Assembler 2.1   07/16/83 19:21:30                  Page 002
parameter list modifier - for BASIC09 or RunB

00062
00063          * prepare limit check for parameter list, allow for ending lastpar
00064    001F 31C900FD                leay   varend-5,u
00065    0023 109F00                  sty    parend5
00066
00067          * copy outname into output parameter list
00068    0026 3410                    pshs   x
00069    0028 3142                    leay   outpar,u
00070    002A 308DFFE3                leax   outname,pcr
00071    002E A680       namechar     lda    ,x+
00072    0030 2704                    beq    nameend
00073    0032 A7A0                    sta    ,y+
00074    0034 20F8                    bra    namechar
00075    0036 3510       nameend      puls   x          input parameter list
00076
00077          * append modified input parameter list into output parameter list
00078    0038 8628                    lda    #'(
00079    003A A7A0                    sta    ,y+
00080    003C 8622                    lda    #'"
00081    003E A7A0                    sta    ,y+
00082    0040 A680       parchar      lda    ,x+
00083          * check resulting parameter list not too long
00084    0042 109C00                  cmpy   parend5
00085    0045 2506                    blo    parOK
00086    0047 53                      comb               set carry
00087    0048 C638                    ldb    #56        BASIC09 parameter error
00088    004A 103F06                  os9    F$Exit
00089          *
00090    004D 8120       parOK        cmpa   #$20       space
00091    004F 2708                    beq    nextpar
00092    0051 810D                    cmpa   #$0d       carriage ret ends parameter li
00093    0053 2712                    beq    lastpar
00094    0055 A7A0                    sta    ,y+
00095    0057 20E7                    bra    parchar
00096          * reformat next parameter
00097    0059 8622       nextpar      lda    #'"
00098    005B A7A0                    sta    ,y+
00099    005D 862C                    lda    #',
00100    005F A7A0                    sta    ,y+
00101    0061 8622                    lda    #'"
00102    0063 A7A0                    sta    ,y+
00103    0065 20D9                    bra    parchar
00104          * list end
00105    0067 8622       lastpar      lda    #'"
00106    0069 A7A0                    sta    ,y+
00107    006B 8629                    lda    #')
00108    006D A7A0                    sta    ,y+
00109    006F 860D                    lda    #$0d       carriage return
00110    0071 A7A0                    sta    ,y+
00111
00112          * fork to interpret (RunB or BASIC09)
00113    0073 308DFFA1                leax   interpt,pcr
00114    0077 108E0100                ldy    #$100      allow one page parameters
00115    007B 3342                    leau   outpar,u
00116    007D 8611                    lda    #prgrm+objct
00117    007F C614                    ldb    #(memory+255)/256 data area
00118    0081 103F03                  os9    F$Fork
00119    0084 2506                    bcs    ut
00120    0086 103F04                  os9    F$Wait
00121    0089 2501                    bcs    ut
Microware OS-9 Assembler 2.1   07/16/83 19:21:49                  Page 003
parameter list modifier - for BASIC09 or RunB

00122    008B 5F                      clrb               no error
00123    008C 103F06     ut           os9    F$Exit
00124    008F 1AA49C                  emod
00125    0092            pgend        equ    *
00126                                 end

00000 error(s)
00000 warning(s)
$0092 00146 program bytes generated
$01CA 00458 data bytes allocated
$1420 05152 bytes used for symbols
```

```
PROCEDURE help_B
REM Dummy Help utility.
REM prints the parameter
PARAM text:STRING
PRINT text
BYE  \REM BYE needed to give automatic return to OS9 when run by BASIC09.
```

UNLOAD..OS9

Steve Childress
31220 La Baya Suite 110
Westlake Vlg. CA 91359

June 1, 1983

'68 Micro Journal

Dear Bit Bucket:

Offered herein is a quickie utility for OS9 level one which is in essence, the inverse of the "LOAD" utility, hence it is named Unload. Users of OS9 are likely all too well acquainted with the frequent need to enter multiple "UNLINK" commands to cause a no longer needed module of code in some language to be removed from memory. The Unload utility therefore is akin to a repeated series of UNLINK commands until the module is indeed removed from memory. Normally, a module is automatically removed from memory, however there are various reasons for a module to be "claimed" more times than unclaimed. When this happens, intentionally or not, the module must be unlinked to removed. The syntax for running Unload from the Shell is:

UNLOAD modulename1 [modulename2 modulenameN]

where those names in [brackets] are optional. Unload is a trivial quickie but none the less, useful.

Enjoy the journal, but Don, would you PLEASE try to get them to do better at paste-ups and use the correct printwheel for language code!

Steve Childress

LISTING for UNload FOLLOWS:

```
UNload              1 Jun 83  20:38 Conejo Computer OS9 Macro Asm. Page
UNload- OS9 Utility


                    ******** Static Storage *******
0000                          ORG     0
0000                stack     rmb     200
          00C8      mem       equ     .

          0011      type      set     prgrm+objct
          0081      revs      set     reent+1

0000                          mod     last,name,type,revs,start,mem
0000 87CD004F000D11816700140 C8
000D 554E6C6F       name      fcs     "UNload"
0013 01                       fcb     1            edition

                    ****************************************************
                    * OS9 Utility to remove a module from memory as    *
                    * would an infinite number of "unlink" commands.   *
                    *      Public Domain: S. Childress; 5/83           *
                    * Synopsis: OS9: UNLOAD modulename [modulename...] *
                    *          >>> An OS9 Level I utility <<<          *
                    * Bugs: "UNload notinmemory" yields no error!      *
                    * Is callable from any program, including SHELL    *
                    ****************************************************

0014                start
0014 327E                     leas    -2,s         workspace
0016 8D2B           a100      bsr     y100         skip spaces within param area
0018 8120                     cmpa    #$20         more names to unlink?
001A 2523                     blo     z800         if no, exit
001C AFE4                     stx     ,s           pointer to module name
001E 4F             a200      clra                 0 => any language/type module
001F 5F                       clrb
                    * Cannot rely on F$link to parse module name because
                    *   UnLoad non-exist module is not an error:
0020 AEE4                     ldx     ,s           pointer to Nth module name
0022 103F00                   os9     F$Link       attempt to link to it
0025 250B                     bcs     a300         if error, stop unlinking
0027 103F02                   os9     F$Unlk       remove module from memory
002A 103F02                   os9     F$Unlk       (note: unlk does not error properly
002D 20EF                     bra     a200         loop 'til link error

002F C1DD           a300      cmpb    #E$LNEM      error-link to non-exist module?
0031 260D                     bne     z900         if no, bomb out
0033 AEE4                     ldx     ,s           param area pointer
0035 A680           a400      lda     ,x+          skip name
0037 8120                     cmpa    #$20
0039 22FA                     bhi     a400         skip name
003B 301F                     leax    -1,x
003D 20D7                     bra     a100         next module, if any
003F
003F 5F             z800      clrb
0040 103F06         z900      os9     F$Exit


                    ****************************************************
                    * Return next non-space char @x
0043 A680           y100      lda     ,x+          skip to next name
0045 8120                     cmpa    #$20         space
0047 27FA                     beq     y100         ignore them
0049 A682                     lda     ,-x          backup to non- pace

004B 39                       rts


UNload              1 Jun 83  20:38 Conejo Computer OS9 Macro Asm. Page    2
UNload- OS9 Utility

004C F83640                   emod
          004F      last      equ     *


 Symbol Table Size:   4662
 3078 Unused Bytes of Macro Storage
    0 Statement Error(s), Last PC: $004E
```

TSC FORTRAN TEST

Route 4, Box 249U
Briarwood Estates
Catlettsburg, Ky 41129
August 26, 1983

Larry E. Williams, Editor
68 Micro Journal
P O Box 849
Hixson, Tennessee 37343

Dear Messrs. Williams:

Enclosed is a printed copy of a brief article on TSC's new FORTRAN 77 subset compiler. I purchased it in May or June but haven't done much with it. Shortly after receiving it, I tried firing it up and ran the 2.0 MHz tests. The article was written on July 4, 1983 and more or less forgotten. A week or so ago I started to putz about with some FORTRAN and found the file. Since I had intended to run tests at 2.5 MHz anyway and had changed the crystal over a month ago, I decided to finish the job. So here it is.

Running at 2.50 MHz has been the least of my problems. At first, I had increasing problems with random resets, but Arnold Comer suggested I check the crystal capacitors (dummy!). That was it. They were 22 pf, supplied with the MP-09D kit. I put in a couple of 18 pf micas and that seems to have completely eliminated the problem. It is running at 2.5 MHz, writing this letter.

The FORTRAN compiler is another story. Since I have just now gotten around to using it, I haven't had any communication with TSC at all on the subject. So far, the only thing it has compiled is the Eratosthenes Sieve programs. I have five other files, several of them quite trivial. The compiler complains that every one of them is too complicated. No other messages. Bah! I have been using FORTRAN compilers since 1964, some of them real duds, a couple very nice, indeed. But never one so totally uncommunicative. Oh, well, it IS a new product.

I ran both of the test program files through the FORTRAN 77 compiler, making printed listings. They are enclosed also. As I promised earlier, I put a new ribbon in the Okidata this time.

Tandy's new sale (?) catalog came Wednesday, hard on the heels of the premiere issue of Color Micro Journal, which came only a couple of days earlier. It looks like the new 64K Color Computer might be some decent competition with a decent looking case, what look like reasonable keys and OS-9. Best Wishes.

Yours very truly,

Wilbur Killebrew Jr.

Wilbur N. Killebrew, Jr.

```
Fortran 77 Compiler 11/15/82
*  FORTRAN 77 Version of Sieve of Eratosthnes
*  Prime-Number "Benchmark"
*  Authors:  J. & S. Gilbreath
*                Byte, January, 1983, page 286
*
*  This test, unlike TSC's SVTEST.TXT, does not
*  depend on the baud rate to the terminal.  The
*  only output produced is a starting message and
*  an ending message.  This program runs the sieve
*  procedure ten times as is specified by Messrs.
*  Gilbreaths' specification.
*
         logical flags(1:8191)
         integer i, prime, k, count, j, iter
*
*  Notify user that test is starting
*
         write(6,900)
900      format(' 10 iterations')
         do 92 iter = 1, 10
           count = 0
           i = 0
*
*  Initialize flags and counter
*
           do 10 i = 1, 8191
10           flags(i) = .true.
*
*  Now find primes
*
           do 91 i = 1, 8191
             if (.not. flags(i)) go to 91
               prime = i + i + 1
*
C              write(6,200) prime
200            format(1x, i6)
*
               count = count + 1
               k = i + prime
               if (k .gt. 8191) go to 91
               do 60 j = k, 8191, prime
60               flags(j) = .false.
91         continue
92       continue
*
*  All done - report total count
*
         write(6,990) prime, count
990      format(1x, i6,' is the largest of ', i6,' primes.')
*
         stop
         end
Fortran 77 Compilation Complete - 52 lines - No errors
```

MORE SIEVE OF ERATOSTHENES BENCHMARKS
TSC's FORTRAN 77

Wilbur N. Killebrew, Jr.

For better or worse, the sieve of Eratosthenes has become a benchmark for microprocessors and their software. The 6800 community has no doubt suffered because of some very slow implementations of BASIC, as far as this test is concerned anyway. One of the files to be delivered with many new compilers is a copy of a version of the sieve. TSC's FORTRAN 77 is no exception.

Besides giving one something to try his new tools on, there is the chance to 'see how well she runs.' Sort of like taking the new '57 Chevy out on a back country road and ...

I had to do it, too. Not bad: 26.0 seconds. But, wait. TSC has thrown us a 'ringer.' This is is not the one published in BYTE magazine. Just to see how the methods compare and to obtain a 100% direct comparasion, I prepared a file containing the Gilbreaths'

TABLE 1

EXECUTION TIMES FOR SIEVE OF ERATOSTHENES BENCHMARK

- Sieve - Version	Ref No.	Language Supplier	Language Name & Ver	O/S	CPU or MPU	Clock MHz	Run Time Secs.
TSC w I/O	(1)	TSC	FORTRAN77	FLEX 1.3	6809	2.000	260.
Gilbreath	(1)	TSC	FORTRAN77	FLEX 1.3	6809	2.000	60.3
Gilbreath	(1)	TSC	FORTRAN77	FLEX 1.3	6809	2.500	48.7
Fixed TSC	(3)	TSC	FORTRAN77	FLEX 1.3	6809	2.000	58.2
Fixed TSC	(3)	TSC	FORTRAN77	FLEX 1.3	6809	2.500	46.8
Gilbreath	(2)	DEC (?)	FORTRAN	?	pdp11/34	?	30
Gilbreath	(2)		FORTRAN	UNIX	pdp11/70	?	15.0
Gilbreath	(2)	Microsoft	FORTRAN3.2	CP/M	Z80	?	13.9

Gilbreath	(2)	Microsoft	FORTRAN	Apple II	Z80		34.
Gilbreath	(2)	Apple	FORTRAN	Apple II	6502		309.
Gilbreath	(2)	Omegasoft	Pascal	-DOS	6809	1.0	40
Gilbreath	(2)	TSC	Pascal	FLEX 9	6809	2.0	54
Gilbreath	(2)	TSC	XBASIC	FLEX 9	6809	2.0	700-963
Gilbreath	(2)	Dig. Res.	CBASIC	CP/M	Z80	?	484.
Gilbreath	(2)	Microsoft	MBASIC 5.2	CP/M	Z80	?	1476.
Gilbreath	(2)	Apple	Applesoft	Apple II	6502		2806.
Gilbreath	(2)	Introl	C	FLEX 9	6809	2.0	11.0
Gilbreath	(2)	BD Systems	C 1.46	CP/M	Z80	?	15.2
Gilbreath	(2)	—	Assembler	—	6809	?	5.1
Gilbreath	(2)	—	Assembler	—	Z80	?	6.8

Notes:

1. Author, using the system described in the text.
2. J. and G. Gilbreath, Byte, January, 1983. Pp 283-320.
3. Author, using his system, with the program 'fixes' described in the text.

FORTRAN IV benchmark, with the appropriate FORTRAN 77 modifications. This was easy to do by just doing a little quick editing of TSC's file, called SVT001.TXT. TSC's sieve program appears at first to be slower than the published 'standard' because theirs runs only once, printing each prime as it is found. Since I/O speed is such an important factor, the 'benchmarks' are run with output disabled. Only two messages are printed by the benchmark: one is printed at the start and another at the end so the test can be timed.

Installing a DO loop takes care of the ten iterations requirement. And it is easy enough to turn off output by converting the appropriate records to comments. But, good grief, look at the size of that command file! Quick, Henry, the reference manual! Somehow, a four sector FORTRAN text file has grown to 106 sectors of command file!

TABLE II

TSC FORTRAN 77 File Sizes

Method	.TXT	.ASM	.REL	.CMD	Notes:
TSC Sieve	4	8	60	106	No LLOAD Z option
TSC Sieve	4	8	64	19	LLOAD Z option
Gilbreath	5	9	69	19	LLOAD Z option

At 106 sectors, the command file obviously contains many FCB's reserving and initializing the program work space. With a working array declared to be 8191 in size, that is a lot of form constants. Fortunately, the new linking loader has a handy option that just fills the bill for bulging diskettes. Z (for Zorro?) zaps the extra zeros, trimming the command file down to a less obese nineteen sectors.

Table I summarizes timing test results. It shows how TSC's new FORTRAN 77 subset compiler stacks up, based on performance with my SWTPCo MP-09B powered system running at 2.0000 MHz. It runs under FLEX 1.3 with [illegible] 1.3. Some readings from BYTE's January, 1983 article by J. and G. Gilbreath are included for comparison.

Just for jollies, I put a 10.0000 MHz crystal in the 'ole MP-09 to see what 2.5000 MHz clocking does for the sieve times. That time is shown in the table also. More on 2.5 MHz operation another time.

All timings were run three to five times and the average taken. This is the number shown in Table I.

W. N. Killebrew, Jr. - August 25, 1983

```
Fortran 77 Compiler 11/15/82
   *  Sieve of Eratosthenes - TSC FORTRAN 77 Version
   *  Modified July 4, 1983 by H. N. Hillebrew for
   *  timing evaluation versus J. & G. Gilbreath's
   *  FORTRAN IV version, BYTE, January, 1983, p. 286.
   *
         logical flags(0:8190)
         integer i, prime, k, count, iter
   *
   *  Run ten iterations for timing tests
   *
         write(6,  1009)
   1009  format(' Sieve of Eratosthenes - start test')
   *
         do 1010 iter = 1, 10
   *
   *  Initialize flags, counter
   *
         do 10 i = 0, 8190
           flags(i) = .true.
   10    continue
         count = 0
   *
   *  Now find primes
   *
         do 100 i = 0, 8190
           if (flags(i)) then
             prime = i + i + 3
             k = i + prime
   20        if(k .gt. 8190) goto 30
               flags(k) = .false.
               k = k + prime
             goto 20
   30        count = count + 1
   *         if(mod(count, 10) .ne. 0) then
   *           write(6, 1000) prime
   *1000       format(i7, $)
   *         else
   *           write(6, 1001) prime
   *1001       format(i7)
   *         endif
           endif
   100   continue
   *
   1010  continue
   *
   *  All done - report total count
   *
         write(6, 1002) count
   1002  format(//1x, i6, ' primes')
   *
         stop
         end
Fortran 77 Compilation Complete - 52 lines - No errors
```

TEXT Decimalization Macros

by Lyle F. Mays
Computer Science Dept.
Missouri Southern State College
Joplin, MO 64801

July 7, 1983

As a computer professional, I have used several text editing and processing systems over the years. While the current text systems tend to favor "on screen" formatting with its dynamic shuffling of text at the operator's interactive demand, there is much to be said for the flexibility and extensibility of the second pass approach as used by classic text processors. These systems often provide such niceties as automatic footnotes, generation of an index and table of contents, and the decimalization formatting especially useful for technical documents.

Decimalization has common application in technical papers where the organization of material is perhaps as important as its content. The basis for decimalization is very similar to that of the outline or hierarchical flowchart. Each level of indentation is denoted by consecutive decimal numbers concatenated to the number describing the next higher level.

For Example:

```
1. Animal
     1.1. Vertebrate
          1.1.1. Mammal
          1.1.2. Non-mammal
     1.2. Invertebrate
2. Vegetable
3. Mineral
     3.1. Solid
     3.2. Liquid
     3.3. Gaseous
```

Happily, the FLEX user has at his/her disposal software which, with the proper extensions, can make possible these and a variety of other special text capacities. One of the most powerful yet least understood pieces of FLEX software is the Technical Systems Consultants Text Processor - better known as PR. Its use of "macros" makes it almost indefinitely extensible, but the implementation of those extensions seems to have been substantially ignored in the confusion of the already complex structure of commands available.

The example of footnoting provided by TSC in PR's documentation is sufficiently complex as to discourage many users from attempting the use of macros other than the simple header and footer types. A minimum of experimentation, however, will show those users some of what they have been missing. The following collection of macros has been designed to illustrate further the ease with which PR can handle the desired text form.

In order to automate the indentation process and the decimal counts which accompany decimalization the following four macros are used: .DR, .DD, .DN, and .DU. These four macros, in addition, define and use the variables Q, R, S, T, U and further call the "sub-macros" .DQ, .DR, .DS, .DT, and .DU which are invisible to the user. All that is required to use the system of macros is to include the following definitions either at the beginning of your text file along with any other text setup (headers, footers, etc.) or, as I do, include them in a MACRO.TXT file as detailed by the PR manual.

The text which is to have decimal indexing applied to it then has the four macros spread through it as follows:

.DR - is usually called only once, at the beginning of the text, as it RESETS all of the levels to zero with the corresponding indentation reset to zero.

.DD - is called each time the level of indentation goes DOWN, that is to a more detailed level which will be made of a longer decimalized label and indented five more spaces to the right. It should be noted that, while the macros only display up to five levels down of decimal labeling, the proper indentation will be carried on to the limit of the line length being indented.

.DN - is called to start the NEXT label at the same level as a previously specified label. The exception to this comes at the beginning of the text - the first label after the reset requires the "next" macro as if to follow an imaginary zero label at the initial reset.

.DU - is called once for each level that you wish to move UP, i.e. back to the left to a less detailed level which would use a shorter decimalization. This call, unlike .DD, doesn't automatically trigger a label to print, since you may need to go up several levels before reaching the desired next label.

To illustrate this process, there follows a three part example: (1) the Decimalization Macro definitions themselves ordinarily included in a MACRO.TXT file, (2) a sample (frivolous) example text file using only those four macros, (3) the output from the previous two parts as run through PR.

```
 1.00=.*             DECIMALIZATION TEXT PROCESSOR MACRO SET
 2.00=.*
 3.00=.*  Provides for indentation and labeling of levels in tech
 4.00=.*  documents.  Currently supports only 5 counters of level
 5.00=.*  positions but carries on with appropriate indentation
 6.00=.*  indefinitely.  These macros use the counter variables
 7.00=.*  Q, R, S, T, and U.  More levels are possible by extend-
 8.00=.*  ing the given definitions but using more variables.
 9.00=.*
10.00=.*  auto increment by ones
11.00=.AU 1
12.00=.* [ .DR ] Decimalization Reset all levels, zero indent
13.00=.DM DR
14.00=.NR Q 0
15.00=.NR R 0
16.00=.NR S 0
17.00=.NR T 0
18.00=.NR U 0
19.00=.IN 0
20.00=..
21.00=.* decimalization level sub-macros
22.00=.*      spacing may be changed as desired SP 2, SP, BR etc.
23.00=.DM QQ
24.00=.SP 2
25.00=@+Q.
26.00=..
27.00=.DM QR
28.00=.SP
29.00=@Q.@+R.
30.00=..
31.00=.DM QS
32.00=.SP
33.00=@Q.@R.@+S.
34.00=..

35.00=.DM QT
36.00=.SP
37.00=@Q.@R.@S.@+T.
38.00=..
39.00=.DM QU
40.00=.BR
41.00=@Q.@R.@S.@T.@+U.
42.00=..
43.00=.* [ .DD ] Decimalization Down one level
44.00=.*      indent & call highest zero level macro
45.00=.DM DD
46.00=.IN @I+5
47.00=.IF @T .IF !@U .QU
48.00=.IF @S .IF !@T .QT
49.00=.IF @R .IF !@S .QS
50.00=.IF @Q .IF !@R .QR
51.00=.IF !@Q .QQ
52.00=..
53.00=.* [ .DN ] Decimalization Next same level
54.00=.*      call lowest non-zero level macro
55.00=.DM DN
56.00=.BR
57.00=.IF !@R .QQ
58.00=.IF @R .IF !@S .QR
59.00=.IF @S .IF !@T .QS
60.00=.IF @T .IF !@U .QT
61.00=.IF @U .IF !@I-20 .QU
62.00=..
63.00=.* [ .DU ] Decimalization Up one level
64.00=.*      remove indent & set lowest non-zero level to zero
65.00=.*      note: this does not cause a level header to print
66.00=.DM DU

67.00=.BR
68.00=.IF @Q .IF !@R .NR Q 0
69.00=.IF @R .IF !@S .NR R 0
70.00=.IF @S .IF !@T .NR S 0
71.00=.IF @T .IF !@U .NR T 0
72.00=.IF @U .IF !@I-20 .NR U 0
73.00=.IN @I-5
74.00=..
```

(2) Sample Run of Decimalization Macros   7-7-83

```
 1.00=.DR
 2.00=THE COMPUTER?
 3.00=.DN
 4.00=Hardware - the part you skin your knuckles on.
 5.00=.DD
 6.00=Main Memory - I forget what it does.
 7.00=.DD
 8.00=RAM - Random Access Memory, a really dumb name? Nothing in
 9.00=a computer is truly random; ROM has the same access as RAM.
10.00=.DD
11.00=Static Devices
12.00=.DN
13.00=Dynamic Devices
14.00=.DU
15.00=.DN
16.00=ROM - would make a good name for a robot.
17.00=.DU
18.00=.DN
19.00=Processor - the part that "processes"?
20.00=.DD
21.00=Motorola 6800 - the past?
22.00=.DN
23.00=Motorola 6809 - the present.
24.00=.DN
25.00=Motorola 68000 - the future!
26.00=.DU
27.00=.DU
28.00=.DN
29.00=Software - doesn't that have something to do with
30.00=Fredrick's of Hollywood?
31.00=.DD
32.00=Operating System - DOS, Monitor, Executive, Master, etc.
33.00=.DD
34.00=Technical Systems Consultants
35.00=.DD

36.00=FLEX
37.00=.DN
38.00=UniFLEX
39.00=.DU
40.00=.DN
41.00=Microware OS-9
42.00=.DU
43.00=.DN
44.00=Languages - Me no speaka da Englesh.
45.00=.DD
46.00=Assembly - for machine efficiency.
47.00=.DN
48.00=BASIC - Beginner's All-purpose Symbolic Instruction Code???
49.00=.DD
50.00=interpreted
51.00=.DN
52.00=compiled
53.00=.DU
54.00=.DN
55.00=Pascal - for programmer efficiency.
56.00=.DU
57.00=.DU
```

(3) Run through PR with above macros yielding the following OUTPUT:

```
THE COMPUTER?


1.   Hardware - the part you skin your knuckles on.

     1.1.  Main Memory - I forget what it does.

           1.1.1.  RAM - Random Access Memory, a really dumb name?
           Nothing in a computer is truly random; ROM has the same
           access as RAM.

                1.1.1.1.  Static Devices

                1.1.1.2.  Dynamic Devices

           1.1.2.  ROM - would make a good name for a robot.




     1.2.  Processor - the part that "processes"?

           1.2.1.  Motorola 6800 - the past?

           1.2.2.  Motorola 6809 - the present.

           1.2.3.  Motorola 68000 - the future!


2.    Software - doesn't that have something to do with Fredrick's
of Hollywood?

     2.1.   Operating  System  - DOS, Monitor, Executive, Master,
     etc.

           2.1.1.  Technical Systems Consultants

                2.1.1.1.  FLEX

                2.1.1.2.  UniFLEX

           2.1.2.  Microware OS-9

     2.2.  Languages - Me no speaka da Englesh.

           2.2.1.  Assembly - for machine efficiency.

           2.2.2.    BASIC   -   Beginner's  All-purpose  Symbolic
           Instruction Code???

                2.2.2.1.  interpreted

                2.2.2.2.  compiled

           2.2.3.  Pascal - for programmer efficiency.
```

As the months pass you will find that 68 MICRO JOURNAL will cater more to the Standard S50 Bus and the growing 68000 users base. COLOR MICRO JOURNAL will direct it's efforts more to the other 6809 systems (more are on the way). There will be some duplication in each publications, however, both will directs it's efforts to a different readership. Readers and advertisers alike will be offered, in the near future, combined incentives to avail themselves to both markets, at reduced rates.

We believe that we return more of our gross income, percentage wise, to our publications, than any other publishing endeavor. In the past ours has been a small market, however, I expect things to change. As we grow so will our size and content. Both in volume and quality.

I sincerely believe that the adoption of Microware's OS-9 disk system and languages is a major event of tremendous proportions in the 6809 community. Ours has been a **QUALITY** base since the beginning, as compared to what the others have offered. Sometimes we have fallen a little behind but we always come back. This is another of those times and **EVERY** one connected with the 6809 community stand to gain, everyone.

From the very earliest days when there was primarily Southwest Technical Products and TSC to look to for support, we survived. Mainly because they were there and supplying state-of-the-art products. Without that support we would not have survived as a group. We are where we are today because they delivered. We all owe them a large debt of gratitude. As the years passed and others joined, things improved. We had wider choices and support.

I have not forgotten and I sincerely hope that you won't either. Our choices are expanding and that is a sure sign of progress and prosperity.

I hope that all of us, readers and advertisers alike will prosper as we more define our market aims, in the coming months. For us, 68 MICRO JOURNAL and COLOR MICRO JOURNAL, as well as S.E. Media and Data-Comp sincerely THANK YOU for all the support in the past and look forward to the future, together.

DMW

- - -

SAGE CUTS PROGRAMMING TIME BY UP TO 90%
==

A new high level programming system, which can reduce programming time by up to 90% has been launched by the London based software house Microprocessor Developments Limited in conjunction with the computer hardware manufacturer Southwest Technical Products.

Called SAGE, the new system can be used develop a complete program package which is more reliable, flexible and presentable than any other item of comparable software on the market. SAGE provides a truly portable environment and operates with UNIX which is rapidly becoming the new standard in multi-user operating systems.

Unlike conventional program generators, which write code in Basic or Cobol, SAGE provides two new high level languages which are designed specifically for modern programming requirements. With these, the amount of code required is reduced by up to 90%, making a program more understandable, reliable and easier to alter if requirements change. Also, as a result of its many powerful, high level commands, programs written with SAGE run some five times faster than equivalent Basic programs.

SAGE consists of four main components:

KEYED FILES: Its keyed files provide an extremely sophisticated direct access system, with retrieval, insertion and deletion times measured in fractions of a second. Unlike many competitive systems, SAGE doesn't have to reorganise its files from time to time.

SCREEN FORM PROGRAM: The user is allowed to create the precise way in which data is presented on the screen with few formal restrictions, yet the process is quite straightforward. Using the powerful screen and data processing language, up to six different files may be viewed and updated at the same time.

REPORT GENERATOR: The purpose designed reporting language can be used to create reports which draw data from several different files. Formatting may be either automatic or user defined, with a wide range of options and full calculation facilities.

UTILITIES: A range of utilities are provided for preparing menus, describing file structures, reformatting files and for defining VDU and printer features.

SAGE has been developed on the Southwest Technical Products range of computers but will soon be available for most machines running UNIX and compatible operating systems. It has already achieved sales in several countries, including the U.S.A., and is receiving acclaim from both distributors and end-users.

SAGE is priced in two packages, the full development system which includes the compilers and a bi-monthly newsletter, and the run-time system required by end-users who do not wish to write their own software.

Ralph Roberts
P O Box 8549
Asheville, NC 28814-8549
(704) 254-3972

AGENT (Books): Judy Segal
15 Sheridan Drive
Pawling, NY 12564
(914) 855-3465

Mr Don Williams Sr, Publisher
'68' MICRO JOURNAL
P O Box 849
Hixson, TN 37343

Dear Don,

The post office reminded me to write you about my minor address change. Please change my subscription address to:

Ralph Roberts
P O Box 8549
Asheville, NC 28814-8549

In other news, I'm still using my SSB Chieftain as a word processor, recently updating from TSC's line editor to John Alford's SCREDITOR III -- which, by the by, I first heard of through 68 MICRO JOURNAL. My book on word processing, 'A GUIDE TO WORD PROCESSING', coming from TAB BOOKS this fall may be of interest to you and your readership. It's 70,000 words and 136 photographs including a color picture of the Smoke Signal Chieftain on the front cover. In it, I review a good many 68xx products and software, including Smoke Signal, Screditor III (good review -- super editor!), Gimix, and a number of others.

My Chieftain has worked well in helping me to achieve a career as a writer. I now have 834 professional sales in National publications. And! My first novel, 'The Bremmertown Berserkers', is currently being filmed as a major motion picture starring Joey Travolta, Pat Paulsen, and Paul Smith for Christmas release.

As to '68' MICRO JOURNAL, I continue to read each and every one. The only complaint, Don, is (**sigh**) the emphasis on FLEX and OS-9. Write about a few DOS articles for a change, huh? Other than that, keep up the good work and all the best of luck in all your endeavors.

From the electronic prose garden.

Ralph Roberts

Ralph: We also would like to 'see' more on SSB DOS. However, I have run everything we have. Seems we do not get much. Guess all SSB users are busy doing other things with their systems.

Now Ralph, to you and other SSB users, I would appreciate some INPUT, then we will HAVE SSB articles to run. How 'bout it?

Thanks for the nice letter and let me know when you have something for our readers. And congratulation on the book. Be looking for an autographed copy.

DMW

- - -

HELP

Owner of a SWTPC 6800 w/SSB disk & DOS68 looking for correspondents with the same for software exchange.
Eric Chomko. 7534 Newberry Lane, Lanham MD 20706.

• • •

HARDWARE/SOFTWARE ENGINEER

Circuit design and programming experience with Motorola family of micro-processors (6802 and 6809). Applicant should have experience in programming for real-time control applications Minimum of 3 years of design experience and a BSEE. or equivalent. required.
Qualified applicants should submit resume to:
Com-Trol, Inc., P.O Box 724, Bucyrus, Ohio 44820. Attn: Carol.

• • •

Wanted Disc Copy of Percom Super Basic for the Percom LFD400 disk system. Will pay reasonable price.
Contact Stanley Lehrer, 36 W. Lenox Ave., Pompton Lakes, NJ. 07442.

• • •

CLASSIFIED ADVERTISING

Elektra CPU (6802) 'Humbug'. $200.00. Thomas S R/R memory board 32K Ram $125.00. Thomas SP 1 1/0 board full $100.00, Gimix 16K Ram card $90.00, SSB 16K Ram card $85 00. SWPTC MP8M 8K Ram $55.00, SWPTC MP4M 4K Ram $30.00. SWTPC MPU (6800), $50.00. 2 Percom LFD400 disk controller cards (MINIDOS+) $75 00 each, 2 Pertec drives w/PS in case $200.00 both. Seriel I/O card $25.00, Dual serial I/O card $35 00, CK7 clock card $35.00. TC3 Cassette card $35 00, MPLS parallel card $15.00. SO Flex BASIC comp./asmb. $150.00. Heath H14 seriel port printer $215.00. Software Inquire.
Write CH(MAJ) Phelps, USMCA/BHR Off. of Comm Chap., APO NY 09034

LIKE NEW - ADDS-VIEWPOINT CRT TERMINALS - DEMO UNITS, THESE ARE LIKE NEW - WARRANTY NEVER REGISTERED - SAVE - ONLY $478.00 EACH - ONLY 2 LEFT. CALL ASK FOR DON, TOM OR LARRY.

A/C 800 338-6800.

**** UNIFLEX Users - SAVE $$$$ ****

New, never Installed or used - High Speed SWTPC **A-1 Tape Back-up System**, 20 megabyte capacity - Complete with new DMF3 Disk Controller - Save over $1,000.00 - **ONLY $2475** Don't get caught without that **BACKUP** for all your data **DON'T MISS THIS ONE!!** - Ask for Don, Tom or Larry, Days

A/C 800 338-6800.

6801 Development System, Motorola Exsorcizer, Dual Disk Drive, 6801 Emulator Terminal. Software and Manuals. $5000
Call John (206) 243-6838.

• • •

Data Systems 6809 CPU. 6845 Video Display, 6847 Video Graphics $65 each. SA-400 Minifloppy with case & PS $100.
Call Joe (617) 587-8241 after 7:30 PM EDT

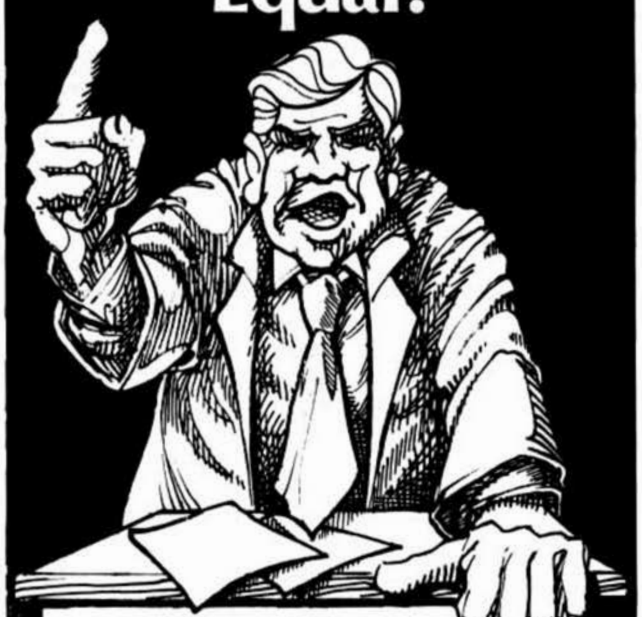
• • •

For Sale: Shugart 5MB SA-604 Winchester drives for $275. very low hours.
Kirk Thomas. 1320 Brunswick Ct., Wichita, Ka. 67212 (316) 722-7507

• • •

For Sale -- SWTP 6800 Computer System. SWTP Boards: MP-A CPU, MP-A2 CPU. 32K RAM, EPROM Programmer, ACIA, 2 - 4K RAM, MP-B Motherboard / Chassis / Power Supply. DATA SYSTEMS BOARDS: 64K RAM DUAL ACIA. JPC Cassette Interface. F&D Disk Controller. SWTP Cassette Interface. SWTP MF-68 Dual Disk System, SWTP CT-64 Terminal w/RS-232 connector. 4K Basic, Cassette Pascal, Pascal Disc, TSC Basic Disk, FLEX, Disc Utilities, Miscelaneous Games on Cassette. CALL (612) 373-2409 days or (612) 722-2921 after 6PM and ask for DAVE.

• • •



For Sale: SWTP DMF-2 controller board - $350 or best offer. SWTP MP-S2 board - $80 or best offer. Both used three months before conversion to S+.
Maurice Leatherbury, 2170 W Kings Hwy., San Antonio, Texas 78201. 512-224-8455.

***

For Sale. GIMIX PIO-28 5in. controller w/flex $200. GIMIX 32K ram cards $85. TANDON 100-4 drives w/ps&case $315. SIEMANS fdd1005 w/ps&case $135. JCP AD-16 $30. HAZELWOOD DM-64 $200. Also lots of flex software.
Bob Randall 414-271-6755 ext 403 days. 414-352 3730 nights and weekends.

***

SHUGART SA4DO $125 each, BASF 6106 SSDD $160 each, S100 8K static ram boards $35 each. M6800 S100 board $50, DATAMEDIA 1521A video terminal $300 ($1100 new); all currently in use, like new, with full documentation, UPS prepaid. Back Issues BYTE - inquire.
Robert Boyd, Woodlawn Avenue, Kennebunkport, Maine 04046.

***

SWTPC 6800 System, DSD 32K card, Percom Disk card, MPI drive, CT-64 video terminal, Thomas Instuments Video card, Keyboard, Data 68 modem card, software, seperate, etc. Name price.
Eric Skarjune, (616)323-0827 after 6, Call Collect.

***

ELEKTRA Cabinet, Power Supply, Disk Reg. List $475 - Sell for $325
William A Comer, Rt 1 Box 437-C, Catlettsburg, Ky. 41179, (606)739-5430

***

COMPILER EVALUATION SERVICES
By: Ron Anderson

The S.E. MEDIA Division of Computer Publishing Inc. Is offering the following "SUBSCRIBER SERVICE":

COMPILER COMPARISION AND EVALUATION REPORT

Due to the constant and rapid updating and enhanchment of numerous compilers, and the different utility, appeal, speed, level of communication, memory usage, etc., of different compilers, the following services are now being offered with periodic updates.

This service, with updates, will allow you who are wary or confused by the various claims of compiler vendors, an opportunity to review comparisons, comments, benchmarks, etc., concerning the many different compilers on the market, for the 6809 microcomputer. Thus the savings could far offset the small cost of this service.

Many have purchased compilers and then discovered that the particular compiler purchased either is not the most efficient for their purposes or does not contain features necessary for their application. Thus the added expense of purchasing additional compiler(s) or not being able to fully utilize the advantages of high level language compilers becomes too expensive.

The following COMPILERS are reviewed initially, more will be reviewed, compared and benchmarked as they become available to the author:

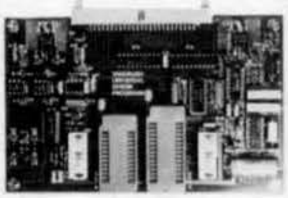
PASCA "C" GSPL WHIMISCAL PL/9

Initial Subscription - $39.95
(Includes 1 year updates)
Updates for 1 year - $14.50

S.E. MEDIA - CPI
5900 Cassandra Smith, POB 794
Hixson, TN 37343
615 842-4601

WINDRUSH MICRO SYSTEMS

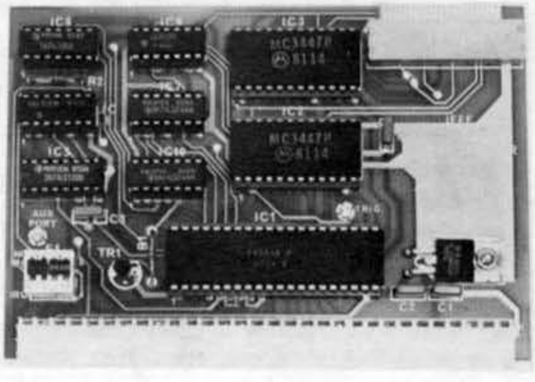
UNIVERSAL EPROM PROGRAMMER



- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified character.
- Software available for all versions of SSB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customizing.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:

  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only...Listen only

- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9

EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTEs and INTEGERs, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (\), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/.XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (XREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: IOSUBS, BITIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCON.

'... THIS IS THE MOST EFFICIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPROC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SP09-17) is REQUIRED.

MACE	(6809 FLEX only)..	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!)............	$198.00
'C'	(a 56K system and the TSC SP09-17 package is req'd)....	$295.00
IEEE-488	with IEEE-488 cable assembly..........................	$298.00
SS-30	Universal EPROM programmer w/one version of software...	$375.00
EXORciser	Universal EPROM programmer w/one version of software...	$395.00
SOFTWARE	Drivers for a 2nd, 3rd, or 4th operating system........	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporations, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES,
NORTH WALSHAM, NORFOLK,
ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97380 SHARET G

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601
For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

---- DISASSEMBLERS ----

Super Sleuth — Powerful, INTERACTIVE, Disassembler; provides Hex/ASCII Screen Dump of Memory Blocks or Disk Files for easy Examine/Change, Area Definitions; X-REF Generator of Source Code; change Address Labels to Names; etc. Disassembles 6800/1/2/3/5/8/9 and 6502 Code.
Color FLEX Obj. Only $50.00, w/source $99.00; **FLEX** $99.00; **UniFLEX** $100.00; **OS-9** $101.00

DYNAMITE + -- An "easy to use" 6809 Disassembler for use w/ Disk Files, (will also disassemble 6800 Code). Develop a "Control File" of Area Definitions during successive disassemblies; X-Ref Source Files; replace Hex locations with Label Names; etc. Label Files provided for Mini-FLEX, FLEX2, FLEX9, Color Computer FLEX. **FLEX** and **Color FLEX** $100.00 **OS-9** $150.00 **UniFLEX** $300.00

—— ASSEMBLERS ——

TSC **Macro Assembler** — the FLEX **STANDARD** Assembler **FLEX** and **Color FLEX** $50.00
TSC **Reloc. Asmb./Link. Load.** — Need for many of the C and Pascal Comp. **FLEX** & **Color FLEX** $150.00

RRMAC — **Relocating, Recursive-Macro Assembler** and **Linking Loader** for the 6809. Use either standard **Motorola** Format or **Special** Ed Smith Format. Supports Recursive Macros, Conditional Assembly, etc. Opt. X-Ref Listing; Includes a Small Line-oriented Editor as part of the Assembler. Greatly improved operating manual. **FLEX** and **Color FLEX** 6809 RRMAC w/Link and Editor $150.00 w/Source, add $50.00

OmegaSoft Relocatable Assembler and **Linking Loader** -- 2-pass Reloc. Asmb; 2-pass Link. Load. Supports 6 Char. Labels, System Calls (SWI[x] FCB x), Expressions with Arith., Logic, and Shifts, etc.
#RAIL1 **FLEX** and **Color FLEX** $125.00 (one year Maint. $50.00)

MACE -- (by Graham Trott) from **WINDRUSH MICRO SYSTEMS**. A combined Editor/Assembler designed to allow the Programmer to Enter, Edit, and Assemble Programs with a minimum of effort, w/o leaving the Program. **XMACE** is a **Cross Assembler** for the MC6800/1/3 and Hitachi HD6301 (CMOS 6801) with the same functions and features as MACE. **FLEX** and **Color FLEX** - $98.00

—— A UniFLEX "basic" De-Compiler ——

DUB -- Re-Create a Source listing from **UniFLEX** Compiled BASIC Programs. Easy to Use; works w/ **ALL** Versions of UniFLEX basic; Output to Disk or Term. Time **TESTED** and **PROVEN**; SOLID! **UniFLEX** $219.95

---- COMPILERS ---

PL/9 -- (by Graham Trott) from **WINDRUSH MICRO SYSTEMS**. A "Structured" Assembly Language Editor/**Compiler**/Debugger, all in **ONE PACKAGE**; provides a totally **INTERACTIVE** Program Development Cycle. The Compiler supports large Symbol Names, Variable Types, Pointers, Control Structures, Stack, A- ,B-, and D-Register manipulation, etc. The **Source** oriented Trace/Debugger provides Single Stepping, Breakpointing, etc. An excellent Software Development Tool for utilizing the power of the 6809.
FLEX and **Color FLEX** - $198.00

C -- (By James McCosh) from **WINDRUSH MICRO SYSTEMS**. SUPER C Compiler for the FLEX Operating System. Needs the TSC Relocating Assembler/Linking Loader for those "full blown" System Packages.
FLEX and **Color FLEX** - 295.00

Introl 6809 "C" Compiler; generates very efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, "68" Micro Journal, May '83). Floats, etc. **FLEX, Color FLEX, OS/9** $375.00 **UniFLEX** $425.00

—— PASCAL ——

TSC **PASCAL** — Native Code Compiler (UCSD Oriented). **FLEX** and **Color FLEX** $200.00

Lucidata PASCAL -- P-Code Compiler (ISO Standard). Designed especially for Microcomputer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Allows linkage to Assembler Code for maximum flexibility.
FLEX and **Color FLEX** 5" $190.00 **FLEX** 8" $205.00

OmegaSoft PASCAL -- For the **PROFESSIONAL**; ISO Based, Native Code Compiler. For Real-Time and Process Control applications. Use custom I/O devices in place of the Pascal INPUT and OUTPUT; Long Int. (32 Bit); Dynamic length strings; Interrupt Proccessing, ROM-able, PIC, Re-Entrant Code, etc. POTENT!! **Includes** Source for the Symbolic Debugger, Runtime, and several Utilities. **Requires** a "Motorola Compatible" Relocating Assembler and Linking Loader.
#PCS2 **FLEX** and **Color FLEX** $425.00 (one year Maint. $100)

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601
For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

—— A full, screen oriented, **WORD PROCESSOR** ——

STYLOGRAPH 2.Ø -- (now runs on the **Data-Comp** and FHL **Color FLEX** Systems; uses the 51 x 24 Display Screens). Full screen display and editing (i.e., what you see is what you get); supports the Daisy Wheel proportional printers.

SPECIAL Color FLEX STYLO $195.ØØ; **FLEX** and **OS-9** STYLO $295.ØØ; **UniFLEX** STYLO $395.ØØ

Fast **SPELLING CHECKER** — allows directly changing the Text File, adding words to the dictionary, etc. 75,ØØØ words in less than 4ØØ sectors. **FLEX, Color FLEX, OS/9** $125.ØØ **UniFLEX** $175.ØØ

MAIL MERGE — greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as differant names and addresses), etc.

FLEX, Color FLEX, OS-9 $145.ØØ **UniFLEX** $195.ØØ

INFOMAG Data Base Management System -- An XBASIC-based, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This **Time Proven** DBMS will become the "Work Horse" of your Software Stable. **FLEX** and **Color FLEX** $295.ØØ **UniFLEX** $395.ØØ

Accts Rec., Accts Payable & Gen Ledger — A **FULL Accounting Package** that can be used together, or as separate packages; provides the IRS required Audit Tracking. (XBASIC, based on the "Osborne Business Programs.") FLEX and Color FLEX $295.ØØ/PROG UniFLEX $395.ØØ/PROG

An **Electronic Spread Sheet**

DYNACALC -- **THE Electronic Spread Sheet** for 6809 Computer Systems. An extremely POWERFUL Business Tool, this Program will find an unlimited number of "non-business" applications, also (for example, I have just finished setting up a Full Junior College Electronics Curriculum using **DYNACALC**). Advanced features like "Table Lookup" make Income Tax work easy; Column or Row Sorting for numerous applications; etc. Completely "Memory Resident", Machine Language, this Program is **FAST**. Provides STANDARD FLEX Text File output for use with BASIC, Word Processors, Pascal, "C", etc.

FLEX and **SPECIAL Color FLEX** (Both FHL and Data-Comp) $2ØØ.ØØ **UniFLEX** $395.ØØ

Machine Language DATA BASE MANAGEMENT System

Westchester Applied Business Systems **XDMS** Data Management Systems. Possibly one of the most powerful DMS's available, this machine language program is **small** enough to operate on a **single sided 5" disk**, yet provides the speed of M.L. and power limited only by the user's imagination. Supports Sequential, Hierarchial, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. Easy-to-use English Language Command Structure.

XDMS — **FLEX** and **Color FLEX** $179.95 XDMS + — **FLEX** and **Color FLEX** $25Ø.ØØ

UNIVERSAL DATA RESEARCH INC. —— Note: **ALL** Accounting and DBM Progs. **Require** FLEX and XBASIC
These are Time Tested programs from an old, established, software house; for **Color FLEX** Systems

Data Base Manager Part 1 - $49.95; **Data Base Manager Part 2** - $49.95
Church Contributions - $49.95 **Single Entry Gen Ledger** - $49.95 **Balanced Billing System** - $49.95

Intergrated Software for **Color FLEX**

A/C $99.95	**A/P** $99.95	**Gen Ledger** $189.ØØ	**Inventory** 2 $69.ØØ	**Payroll** $99.95

FLEX and **UniFLEX** — Note: Requires **XBASIC** (FLEX) or **basic** (UniFLEX)

A/C	FLEX $295	UniFLEX $395	**A/R**	FLEX $295	UniFLEX $395
Gen Ledger	FLEX $295	UniFLEX $395	**Inventory 2**	FLEX $295	UniFLEX $395
Payroll	FLEX $295	UniFLEX $395	**DBM**	FLEX $35Ø	UniFLEX $45Ø

Please specify 5 or 8 inch disk when ordering all software!

Computer Systems Consultants **FLEX XBASIC** Programs

FULL SCREEN FORMS DISPLAY	**FLEX** and **Color FLEX**	$5Ø.ØØ	**UniFLEX**	$75.ØØ
FULL SCREEN MAILING LIST	**FLEX** and **Color FLEX**	$1ØØ.ØØ	**UniFLEX**	$11Ø.ØØ
FULL SCREEN INVENTORY/MRP	**FLEX** and **Color FLEX**	$1ØØ.ØØ	**UniFLEX**	$15Ø.ØØ
TABULA RASA SPREADSHEET	**FLEX** and **Color FLEX**	$1ØØ.ØØ	**UniFLEX**	$2ØØ.ØØ

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601
For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

SPELLB "Computer Dictionary" ---- **OVER 120,000 words!**

No more "Let your fingers do the walking through the Dictionary" while you are inputting Text with your favorite Editor or Word Processor. **SPELLB** is more than "another Spelling Checker"; it allows you to **look up a word** from within your Editor or Word Processor so that you **KNOW** it is right **WHEN** YOU TYPE IT IN with the **SPH.CMD** Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View a word in context" before changing or ignoring, etc. **SPELLB** first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. **SPELLB** also allows the use of **Small Disk Storage** systems.

FLEX and **Color FLEX** $129.95

JUST --- a Text Formatter

JUST, a Text Formatter developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. **JUST** is designed for formatting Text Output for **Dot Matrix Printers** and provides many unique features:
-Output the "Formatted" Text to the Display for format analysis and change.
-Output the "Formatted" Text to a Text File for use with the **supplied FPRINT.CMD** for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility is very useful at other times also, and worth the price of the program by itself).
-"**User Configurable**" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, boldface on and off, etc.
-Automatic compensation for a "Double Width" printed line.
-Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.
-Use with **ANY** Editor.
-Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

FLEX and **Color FLEX** $49.95

SPECIAL!! SPECIAL!! SPECIAL!!

Star-Kits excellent **SPELL'N FIX Dictionary** and **WRITE 'N SPELL** Word Look Up Program IN ONE PACKAGE;
FLEX and **Color FLEX** Systems ---- **BOTH for ONLY $150.00**
When these are gone; the price goes UP!! WAY UP!! ORDER NOW!!

Also, call for "More Info" on both the FLEX Based and Color Computer Based **STAR-Kits** Products; including the HUMBUG Monitor, Check 'N Tax Program, REMOTERM Color Computer External Terminal Program, etc.

PASCAL UTILITIES — Requires LUCIDATA Pascal ver 3.

XREF — produce a Cross Reference Listing of any text; oriented to Pascal Source.
INCLUDE -- allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions.
PROFILER -- produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

FLEX and **Color FLEX** -- Each program $25.00

COPYCAT -- (Pascal **NOT** required) Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in **Modular Source Code** to make it easier to solve unusual problems.

FLEX and **Color FLEX** 5" $50.00 **FLEX** 8" $65.00

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

O-F —— OS/9 to FLEX - FLEX to OS/9 ——

Finally; the barrier has been removed between OS/9 and FLEX formatted disks! Now you can READ from, and WRITE to, a Single Sided 5" or 8" FLEX diskette from OS-9 with **O-F**. **O-F** is a new and unique program, written in **BASIC09** (**with** Source), that performs the following functions;

REFORMAT: A **BASIC09** Program that reformats a chosen amount of an OS-9 disk to **FLEX** Format so it can be used normally by FLEX.

FLEX: A **BASIC09** Program that does the actual **read** or **write** function to the special **O-F Transfer Disk**, all selectable from a user-friendly menu. Functions provided include reading the FLEX Directory, Deleting FLEX Files, Copying both directions, etc. All selections are interactive and complete, including all necessary prompts to the operator.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

FLEX and **Color FLEX** $79.95

COPYMULT.CMD --- Copy LARGE Disks to several smaller disks ---

The following FLEX utilities allow the backup of **ANY size disk** to any SMALLER size diskettes (Winchester to 8's or 5's, 8" to 5's, etc.). By simply inserting diskettes as requested by **COPYMULT**, a large disk system may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using the normal copy routines.

COPYMULT.CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP.CMD is a special program that downloads "random" type files, any size.

RESTORE.CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK.CMD a "bonus" utility that "relinks" the free chain of floppy or hard disk thereby eliminating fragmentation.

Completely documented source files included. **ALL 4 Programs** $99.50 (8" or 5")

CHESS 6809

Requires FLEX and DISPLAYS **On Any Type Terminal**

Features:

*Two display boards. *Change skill level. *Swap side. *Point scoring system.
*Four levels of play. *Solve Checkmate problems in 1-2-3-4 moves.
*Make move and swap sides. *Play white or black.

This is one of the **strongest** CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels).

FLEX and **Color FLEX** $79.95

DIET-TRAC Forecaster

DIET-TRAC Forecaster is an XBASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate. Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

FLEX - $59.95 **UniFLEX** - $89.95

XDATA —— A COMMUNICATION Package for the UniFLEX Operating System ——

Allows **UniFLEX Based Systems** to Transmit and Receive files to and from other Computer Systems via Modem. Use with CP/M, Main Frames, other UniFLEX Systems, etc.

— Verifies Transmission integrity using checksum or CRC
-- Automatically Re-Transmits bad blocks
-- Transmits data in 128 byte blocks

UniFLEX $299.99

*FLEX is a trademark of Technical Systems Consultants
**OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SOUTH EAST MEDIA

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601
For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

AT LAST!! A FULL BLOWN DISASSEMBLER FOR THE COLOR COMPUTER

Computer Systems Consultants **SUPER SLEUTH** is a "Time Tested", reliable, **PROVEN** Disassembler that has gained acceptance through out the FLEX Community as an extremely POWERFUL, **INTERACTIVE**, Software Tool. Now, this powerful Disassembler has been converted to run on a **Standard 32K Color Computer** or TDP-100 System with a **Disk System**. The CoCo **SLEUTH** Software Package consists of 3 Programs; SLEUTH (the Disassembler), CHGNAM (used to globally Change Labels to a meaningful Name), and XREF (a Cross Reference Generator for Source Code Files). CoCo SLEUTH will Disassemble Disk Files of 6800, 6801, 6802, 6803 (the "Baby CoCo"), 6805, 6808, 6809, and 6502 (Apple, Atari, Commodore, etc.) Object Code if you can get it on a Color Computer Disk. (See Aug. '83 **'68' Micro Journal** "Color Users Notes" Column for a full Review.)

Color Computer Disk - Object Code Only $49.00

FORTH Programming Language

Stearns Electronics FORTH -- Intrigued by **Forth**??? Here is a Forth package **tailored** to the **Color Computer**! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's **speed is unparalled**. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning Forth, a **Trace Feature** is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of this Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

Color Computer TAPE (w/ instructions for transferring to Disk) $58.95

Color Computer GRAPHIC SCREEN PRINT Programs

Dumps any "PMODE" Screen to the Printer with the BASIC USR Function. Shift the Printout Left or Right or Reverse Print (Dark for Light Screen and Vice Versa). All Programs on Tape.

GSPR	for Radio Shack LP-VII/VIII & DMP 100/200/400 Printers	$7.95
GSPRE	for Epson w/ Graftrax and Graftrax + Printers	9.95
GSPRG	for Gemini 10 and 15 Printers	9.95
GSPRP	for the Prowriter Printers	9.95

DATE-O-BASE CALENDAR Program

A Menu Driven **EXTENDED BASIC** Program which allows the entry of up to 12 Memos per Day, each of which may contain up to 28 Characters, for any day of the Month between the years 1700 and 2099. A **Graphic Calendar** shows which days contain Memos, and a "Key Word" Search is provided which can be output to the Screen or Printer.

TAPE DATE-O-BASE CALENDAR (Each Tape File will hold up to 400 Memos)	$16.95
DISK DATE-P-BASE CALENDAR (4,000 Memos at 300/Month per Disk)	19.95

Interested in INTEREST (the Money Kind)?

An **EXTENDED BASIC** Program that will help you deal with numerous problems requiring interest calculations. Present Value, Rate of Return, Current Bond Yield and Rate of Return to maturity, Loan Repayment Amortization Scheduals, etc.

TAPE $29.95

Data Base Management System

DISK DATA HANDLER 64K - EXTENDED BASIC w/ Mach. Lang. Routines. Allows a max of 246 Chars. and 14 Fields per Record, and another Record can be linked to the first; 8 Char. Field Names, up to 99 Chars. per Field. Powerful On-Screen editor for input and update, flexible Output capabilities including output to Disk Files for use by other Programs. Change File Definition without re-entering the Data, Split Files, etc. Allows Multiple Field Sorts, Select on any combination of Fields, etc. An extremely POWERFUL **TOOL**; instructions provide examples of Mailing Lists and a Financial Stock Profit and Loss Tracking System.

DISK $54.95

ACCOUNTING

DISK DOUBLE ENTRY - DISK EXTENDED BASIC w/ Mach. Lang. Routines. A "Traditional" Accounting Package for Small Business, Clubs, Churches, Personal Use, etc. Up to four levels of subtotals with Trial Balance, Income Statement, and Balance Sheet Reports. DDE allows up to 300 accounts and a Trial Balance of $9,999,999.99. Transactions may be up to 14 lines long, and comments and explanations may be freely used. Accounts are traceable to the journal transaction, which may include comments. Screen reports allow review of past transactions and current balances.

DISK $44.95

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601
For Ordering Call TOLL FREE 1-800-338-6800

Software

FLEX™ OS-9™ Color Computer

DYNASHARE — Multi-User, Multi-Tasking with FLEX

Southeast Media is now shipping DYNASHARE **FROM STOCK** - the multi-user, multi-tasking capability of DYNASHARE allows FLEX users the advantages of more sophisticated and time saving computer usage without having to buy or learn a new Language or Operating System syntax. **DYNASHARE**, as its name implies, allows true "time-sharing" operation under the popular FLEX operating system, and also allows each user to run two simultaneous jobs (multi-tasking); even on single-user systems. For example, while in EDIT, you can list another file or examine a directory. Or, you might look up an item in a Data Base while a Sort is in progress! **DYNASHARE** also provides some fringe benefits that will be greatly appreciated by FLEX users, including type-ahead, command line editing, and instant response to "escape".

DYNASHARE is the painless method! Use your existing Flex computer by simply adding 64K of RAM for each user. Fact is, you still use FLEX just like you always have! **DYNASHARE** is not intended as competition to UniFLEX. It does not improve on the speed of FLEX, and does not offer password protection or other niceties of a full-blown multi-user system. What **DYNASHARE does** do is give FLEX users a **low-cost way to use existing software in a multi-user, multi-tasking environment**, so your existing FLEX versions of BASIC, XBASIC, editors, assemblers, disassemblers, sort/merge packages, word processors, compilers, DYNACALC spread-sheet package, and so on are still good.

NOTE -- The **initial release** of DYNASHARE is for **SWTPC S/09** Computers, but versions will also be available for other popular extended-memory (up to **1024K**) systems, such as HELIX and GIMIX. A minimum of 128K of RAM will be required with **ALL** versions. **DYNASHARE** requires **64k of RAM for each active task**; thus a 256k system could allow foreground-background operation on two terminals, or foreground-only operation on four terminals. **AVAILABLE NOW** from **SOUTHEAST MEDIA** --- $200.00

--- AUTHORS - PROGRAMMERS ------ QUALITY SOFTWARE NEEDED ---

FLEX - UniFLEX - OS/9 - Color Computer

For the past several months, we at the **Southeast Media Division** of **Computer Publishing, Inc. (CPI)**, the parent company of **'68' MICRO JOURNAL** and **COLOR MICRO JOURNAL**, have debated expanding our software distribution business. Many other magazines have been doing so for years (in fact, MOST were in the Software Distribution Business BEFORE they began to publish a Magazine). Presently there are many fine examples of software that has been developed by YOU, our readers, that will never see the "light of day" due to the cost of Advertising and TIME and cost involved in the production, distribution, and Customer SUPPORT of that software unless SOMEONE, with enough exposure and the willingness to continually advertise, runs with the ball.

Software is the "backbone" for the REAL utilization of any Computer System, and ours are no exception! This has been no simple decision. While we realize that there could be some conflict with some of our advertisers, we ALSO hear a LOUD and CONTINOUS cry for **HELP** from our Readers. From day one, the **foremost concern** of '68' MICRO JOURNAL has been it's **READERS!** Therefore, our **Southeast Media Division** will accept, for appraisal for possible Distribution, 6809 software; Games, Utilities, Software Development, Business Application Programs, etc.

In the past there has been **too much** software offered that was not quite ready. We will strive to eliminate that element. But, right up front, we tell you only that we will do our very best; nothing more. Also, we will strive to keep cost to a bare minimum, while securing for the author a fair return in royalty payments, promptly paid, and in customer support for his product.

Of course, we will expect, no -- **DEMAND**, that the author keep the product free of errors (bugs), and maintain it in a prompt and business like manner. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a "single" sale product. In this event it will be combined with other like programs, and offered as a package. In that event, the royalties will be split between the various authors.

If you have software that you feel will qualify under this program, please contact one of the people below. Remember, if your software has any problems or "funnies" -- **GET IT STRAIGHT BEFORE YOU CONTACT US!!** Also get your source code in proper shape and well commented; there is too much 99% code already drifting around.

If your software is **READY** contact: **Bob Nay**, Don Williams, or Tom Williams

Southeast Media is a division of **Computer Publishing, Inc. (CPI)**,
a family of 100% 68XX support facilities.

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Add $2.00 Per Item For Shipping & Handling

For Ordering Call TOLL
FREE 1-800-338-6800

FREE DISKETTE With Each $50 Purchase

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

S
A
L
E

S
A
L
E

S
A
L
E

*** ALFORD'S GOOD NEWS ***

PRE-INVENTORY CLEARANCE SALE!!

Its inventory time around here. To make things easier on ourselves, we want to sell as much of our current inventory as we can. So until stock is depleted, we are offering some FANTASTIC DEALS on selected items. Remember, these prices are ONLY good until our current stock is gone, so call or write to place your order today!

VS-1 SPEECH SYNTHESIZER.....$120

This is the most popular unlimited speech synthesizer available for the SS-50 I/O bus! All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. If you buy this unit while on sale, we will include a copy of our SC-1 text translator program at no additional charge. This gives you a $170 value. You save $50!! Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating system and disk size when ordering.

SP-1 SPEECH SYNTHESIZER.....$110

Our popular unlimited speech synthesizer for the Radio Shack Color Computer is also available on this sale. All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. Buy your SP-1 on this sale and we will INCLUDE our SC-1/COLOR translator program which lets your CoCo speak English using nothing more than PRINT statements! Also, we will throw in our COLOR-TREK+ program. This is our COLOR-TREK program adapted for the SP-1. English messages are continually spoken during game play! The action is non-stop. This makes a $170+ value. You save 50+! Software is available on CASSETTE ONLY.

SC-01A SYNTHESIZER IC'S.....$ 55

These ohips are the ones used in our synthesizer boards, and are prime, new parts directly from Votrax. We have a limited number of this most popular speech synthesizer I.C. in stock for this sale. Suggested retail price is $85. Discount houses are selling them for $70. You save $15 on the discount house price. We include copies of the Votrax spec sheets with these parts.

SCREDITOR-III.................$150

All of our customers who have used both of the other major word processing packages available for 68xx systems tell us that SCREDITOR III is the best program that they have ever seen (in truth, ONE customer liked one of the others better, but you can't please EVERYONE!). As of July 15th, SCREDITOR III will cost $175. If you have been considering a word processor, or just need a better editor than what you are using now, this is the time to buy SCREDITOR III! For more information about this fantastic program which includes over 150 pages of documentation (and another 240 pages coming soon!). see our ads in the past several issues of Micro-Journal. Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating systems and disk size when ordering.

ORDERING INFORMATION...

All sales are FINAL. No returns or refunds, NO open account orders and NO DEALER DISCOUNTS on VS-1, SP-1 or SC-01A purchases for the duration of this sale. Normal dealer discounts and open account terms will apply to SCREDITOR III purchases.

All overseas orders MUST include $10 for shipping. All North American orders under $100 MUST include $5 for shipping. ALL checks MUST be drawn in U.S. dollars on accounts in U.S. banks. No foreign banks or currencies will be accepted under ANY conditions. All Virginia orders MUST include 4% state sales tax. COD orders in the U.S. ONLY, and must be shipped by UPS. All overseas orders shipped by AIR PARCEL POST. We gladly accept VISA and MASTERCARD orders.

Please allow two to three weeks for order processing. Orders paid by check may take longer. Stocks are limited. Any orders received after stocks are exausted will be returned unprocessed.

ALFORD AND ASSOCIATES
PO BOX 6683
RICHMOND VA 23230
PH 804-320-6722

FLEX is a trademark of Technical Systems Consultants. OS-9 is a trademark of Microware Systems Corporation.

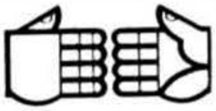
The Lord is our primary associate. Too often with the pressures of business today, time is not taken to acknowledge His presence, and too often we fall, due to our human weaknesses, to show His influence in our lives. We thank each of our customers for their patience and prayers for us when we do not reflect Christ in our business activities.

S
A
L
E

S
A
L
E

S
A
L
E

CONGRATULATIONS, ARTISAN!
CONGRATULATIONS, ELEKTRA!



When Artisan Systems Corporation needed a system monitor for their dual-processor single-board computer, where did they turn? To Star-Kits, of course!

When AAA Chicago Computer Center needed a system monitor for their 6800 and 6809 computers, where did they turn? To Star-Kits, of course!

We would like to welcome these progressive manufacturers to the world of Star-Kits software, and thank them for adopting our HUMBUG monitor for their products. We have recently concluded several licensing agreements for HUMBUG, and for our STAR-DOS disk operating system, and there are more in the works. We're glad that other manufacturers think enough of our products to put them into *theirs*.

What about you? Maybe it's time for you to put our software into your hardware too. Send for our catalog.

STAR KITS

P.O. BOX 209 — J
MT. KISCO, N.Y. 10549
(914) 241-0287

OS9
APPLICATION SOFTWARE

ACCOUNTS PAYABLE $299

GENERAL LEDGER with CASH JOURNAL $399

PAYROLL $499

ACCOUNTS RECEIVABLE $299

SMALL BUSINESS INVENTORY $299

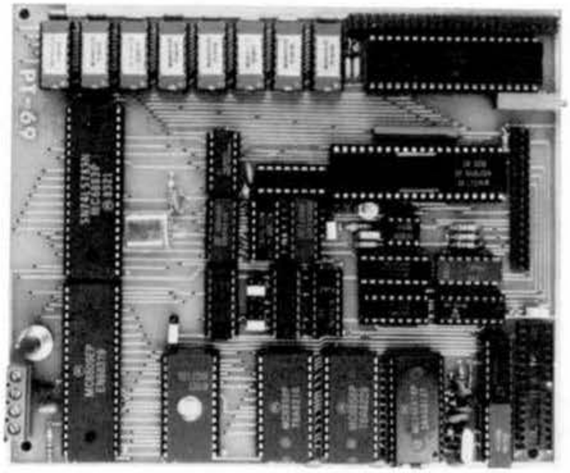
COMPLETE DOCUMENTATION $19.95

OS9 & BASIC 09 ARE TRADEMARK OF MICROWARE, INC. & MOTOROLA CORP.

SPECIALTY ELECTRONICS

(405) 233-5564
2110 W. WILLOW — ENID, OK 73701

6809 SINGLE BOARD COMPUTER 6809
ADVANCED TECHNOLOGY ECONOMICAL



BOARD SIZE
5½" x 6½"

The new PT-69 is a complete computer system on a single card. Compact, powerful, with performance equal to bigger systems. The PT-69 features:

- 6809E Processor
- Double Sided/Double Density Floppy Disk Controller
- 2 RS232 Serial Ports (6850)
- 2 8-Bit Parallel Ports (6821)
- Time-of-Day Clock (MC146818)
- Power Requirements are +5 @ 800MA and ±12 @ 20MA
- Run FLEX 9 or SWTPC FLEX
- 56K RAM: 4K EPROM: 4K I/O
- LOW COST:

• PT-69, Assembled and Tested	$295.95
• PT-69, Assembled, Tested, with Power Supply and Cabinet	$374.95
• COMPLETE SYSTEM with PT-69 Board, 2 DS/DD 5¼" Drives, Cabinet, and Power Supply.	$895.95

— Kits are available -
Write for Details!

PERIPHERAL TECHNOLOGY
"Supplying Your Computer Needs Since 1978"
3760 Lower Roswell Road
Marietta, Georgia 30067
VISA/MASTERCARD/CHECK/COD 404/973-0042

™FLEX is a trademark of Technical Systems Consultants

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. What is an electronic spread-sheet, anyway?
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. Is DYNACALC just for accountants, then?
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. What will DYNACALC do for ME?
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. Do I have to learn computer programming?
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. Do I have to modify my system to use DYNACALC?
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

6. Will DYNACALC read my existing data files?
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. How fast is DYNACALC?
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. Is there a version of DYNACALC for MY system?
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. How much does DYNACALC cost?
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. Where do I order DYNACALC?
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Mo orola.

'68' MICRO JOURNAL

★ The only ALL 6800 Computer Magazine.

★ **More** 6800 material than all the others combined: **MAGAZINE COMPARISON**

(2 years)

Monthly Averages

6800 Articles

KB	BYTE	CC	DOBB'S '	TOTAL PAGES
7.8	6.4	2.7	2.2	19.1 ea. mo.

Average cost for all four each month: $6.53
(Based on advertised 1-year subscription price)

'68' cost per month: $2.04

That's Right! Much, Much More
for About
1/3 the Cost!

- - - - - - - - - - - - - - - - - - - -

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Master Charge □ — VISA □

Card #.______________ Exp. Date__________

For □ 1-Year □ 2 Years □ 3 Yea s

Enclosed: $______________

Name______________________________

Street______________________________

City______________ State__________ Zip______

My Computer Is:______________________

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

SUBSCRIPTION RATES

USA
1 Year $24.50, 2 Year $42.50, 3 Year $64.50

*FOREIGN SURFACE Add $12.00 per Year to USA Price

*FOREIGN AIRMAIL Add $36.00 per Year to USA Price

**CANADA & MEXICO Add $5.50 per Year to USA Price
Cash (USA) or drawn on a USA Bank!!!

ARTISAN SYSTEMS CORP.

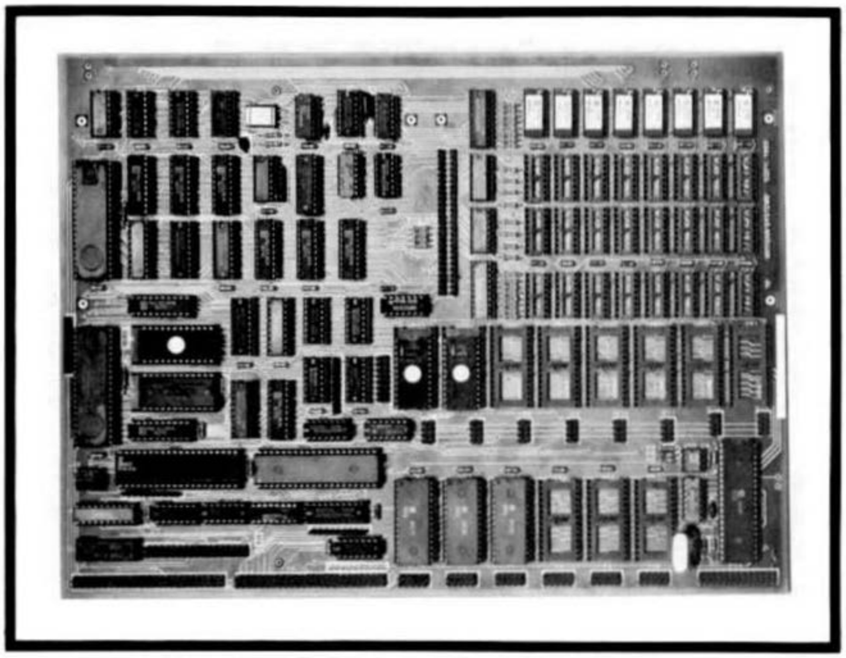
DP-09
6809 BASED SINGLE BOARD SYSTEM

2 MHZ OPERATION

256K DRAM

6 RS-232 SERIAL PORTS

8 28-PIN SOCKETS



DUAL 68B09E PROCESSORS

FLOPPY CONTROLLER

WINCHESTER INTERFACE

4 LAYER PC BOARD

FEATURES:

- 68B09E advanced 8/16 bit system processor with memory management hardware allows for one megabyte address space.
- 64K - 256K byte DRAM.
- 8 each 28-pin sockets for up to 128K byte EPROM. EPROMS can be 2732, 2764 or 27128. In addition 2K x 8 or 8K x 8 static RAMS may be used.
- Six RS-232 serial ports with full modem handshake. Advanced 6551A ACIAs with software baud rate select of 110 to 19.2K baud.
- 6522A interface chip provides two 16-bit timers plus two 8-bit parallel ports (unbuffered)
- 8 auto-vectored interrupts for high speed I/O handling
- 50-pin expansion socket
- Independent 68B09E subsystem for disk control 1-4 floppys 5" or 8" DS, SD, DD. SASI interface allows 5-45 MBYTE Winchesters to be connected using external controller. The subsystem uses a proprietary DMA technique for high speed operation.
- FLEX operating system is available.

PRICES

DP-09A & T 64K, 4 Serial Ports	**$895**
FLEX for DP-09....................	$150
OS-9 Level One	CALL
Forth	CALL

ARTISAN SYSTEMS CORP.
410 CROSS STREET
WINCHESTER, MA 01890
(617) 721-2109

ALL ORDERS PREPAID. VISA & MASTERCARD ACCEPTED. 3-4 WEEKS DELIVERY. (ADD 2 WEEKS FOR PERSONAL CHECKS)
FLEX™ is a trademark of Technical Systems Consultants. OS-9™ is a trademark of Microware.

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

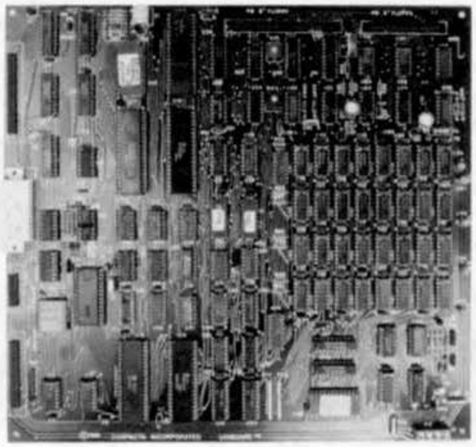
64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!

BLANK PC BOARD

$\$99^{95}$

WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.



$\$399^{00}$

COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful Instruction set available today on ANY 8 bit micro. Now, at last, all of that Immense computing power is available at a truly unbelievably low price.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input Interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion Interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

64K SS-50 STATIC RAM

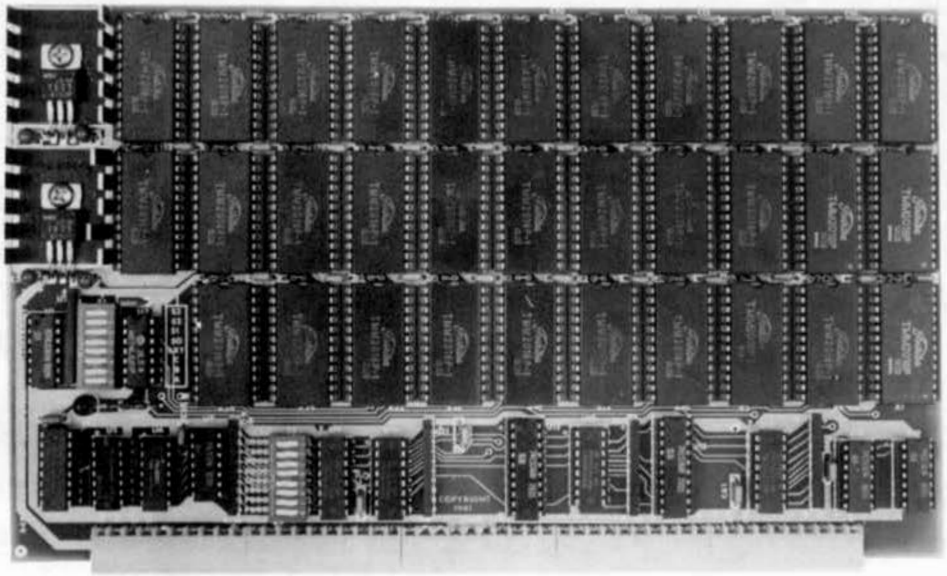
$179.00 (48K KIT)

NEW!

NEW!

LOW POWER!

RAM OR EPROM!



BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (wi hin any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

CLOSE OUT SPECIAL

WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

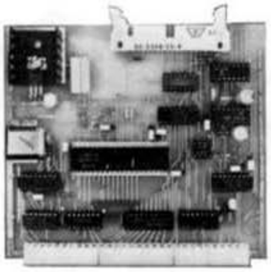
The new 2K x 8. 24 PIN. static RAMs are the next generation of high density, high speed, low power. RAMs. Pioneered by such companies as HITACHI and TOSHIBA. and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts. feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality. and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2 00 postage We pay balance Order under $15 add 75¢ handling No C O D We accept Visa and MasterCharge Tex Res add 5% Tax Foreign orders (except Canada) add 20% P & H Orders over $50. add 85¢ for insurance.

DOUBLE DENSITY OPERATION FOR 6800 AND 6809 COMPUTERS

DOUBLE YOUR STORAGE CAPACITY



SS-30 or SS-30C

Now you can run double density on any S-50 6800 or 6809 computer. Who else can match these features at these low prices! The FD-2 features:

- Control up to four 5¼" DS/DD Drives
- SS-30 or SS-30C compatible
- Uses the TSC or SWTPC FLEX Operating System
- 2.0 MHz operation with no "slow I/O required"
- Run double density on 6800 systems having a 894 KHz clock or faster
- Compatible with SWTPC DC1, DC2, DC3, or DC4 controllers

FD-2	Assembled and tested controller card	$149.95
FD-2S	6800 Double Density Drivers + Format Program	$ 19.95
FD-2DSDD	FD-2, 2 40 TR Double Sided Drives, Cabinet Power Supply, and Cables	$749.95
FD-1A	CLOSE OUT SPECIAL! FD-1 Single Density Controller	$ 49.95

PERIPHERAL TECHNOLOGY

"Supplying Floppy Disk Controllers Since 1978"
3760 Lower Roswell Road
Marietta, Georgia 30067
VISA/MASTERCARD/CHECK/COD 404/973-0042

TMFLEX is a trademark of Technical Systems Consultants

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display procesor. High resolution 256 x 192 pixel display with 15 colors. 16K Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allows subtitling. NTSC composite video output.

SOUND EFFECTS AND MUSIC

- Three AY3-8910 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- Onboard stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES

- Eight analog inputs with 8 bit resolution
- Supports four joysticks with pushbutton switches
- Eight bit parallel I/O port
- Entire unit maps into 25 bytes of memory

FBASIC

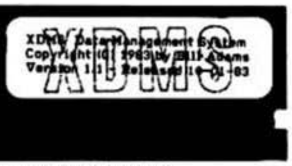
TERMINUS DESIGN INC. in conjuntion with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability Graphics verbs and functions fully support the Arcade 50.

ARCADE 50 as mbled and tested	$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbus)	375.00
NEW MV09 6809 Processor Board	225.00
256K Dynamic Memory Board	795.00
256K Dynamic Memory Board (w/64K	395.00
64K Dynamic Memory oard	295.00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

TERMS: CASH, VISA, MC, COD

LOOK FOR THIS LABEL!



XDMS Data Management System
Copyright (C) 1983 by Bill Adams
Version 1.1 - Released 10-01-83

Note! Pre-8/83 shipments do not have the boldface "XDMS"

This label is your assurance that you have an ORIGINAL copy of the XDMS system. It is also your Proof of Purchase which entitles you to technical support, notification of changes and enhancements, and system upgrades at reduced rates (Version upgrades are $25.00 plus shipping).

We register all XDMS sales in our user database (implemented under XDMS, of course). This provides us with both a mailing list, and an inventory of users. We know who purchased what version and when. By comparing dates with our modification log, we can tell you what changes affected your system, and what has occurred since you purchased your system. When applicable: we will recommend a later copy within a version and will update your disk for a small service fee of $5.00 plus shipping. ($10.00 if we supply a new disk).

We are sensitive to your needs. If you require a specific function - and it is GENERIC - we will investigate implementation. We are USER DRIVEN! XDMS owes much of its flexibility to user requests. In the first six months of circulation, eight enhancements were at the request of users.

XDMS is personalized to you or your business. Your name or business name appears as part of the licensing and copyright messages which appear when XDMS is invoked. If yours is missing or incorrect, you have a bogus copy - contact us with your supplier's name - He owes us money and owes you an original copy! (Note: Authorized dealers are permitted to loan out their copies PROVIDED THAT your order has been placed with us!)

These measures are to assure you - the end user - of the highest quality and standards possible. We believe in supporting our product. We dislike getting surprise phone calls from unknown users (which has happened). We fully r[illegible] our product and believe it's the best available. In fact, XDMS has evolved over three years to rival DBMS's available for other (non-68XX) systems which cost $500. and up. You know who we mean. We believe the '68 family of processors is THE BEST, and will continue to provide the most flexible, easy-to-use software possible.

Should you "look for this label?". We think so. Because it stands for more than just a good piece of software. It stands for quality, for support, and above all, for a community of users who may pride themselves with excellence in their selection of both hardware and software...

WESTCHESTER Applied Business Systems
Post Office Box 187
Briarcliff Manor, N.Y. 10510

XDMS DATA MANAGEMENT SYSTEM

The XDMS (eXtended) Data Management System goes beyond the traditional functionality offered by database management systems. It is capable of the normal tabular data manipulations, calculations, sorting, searching and report generation. XDMS adds to these basic features the ability to intermix file data with text to produce custom invoices, contracts and personalized business letters on a mass production or selective basis. XDMS defines a relational structure whereby two files may be linked together via a third. This aspect may be employed to correlate report data on a many-to-one-to-many basis. Advanced record selection permits 'driving' reports and inquiries with a file derived from a secondary selection process. All this functionality is available within a single process, controlled by a single user written English-like control file.

XDMS supports sequential, hierarchical and random file structures and employs data compression to permit large files on limited disk space. The hierarchical format further reduces space re uired. If custom processing or file transfer are required, XDMS files may be easily translated to text format. XDMS files may be defined with up to 24 fields and record lengths up to 255 bytes. Alphanumeric, numeric, decimal, integer, scalar date, coded and hexadecimal field types are supported. Potential keys to other files are simply defined by similar field names, types and sizes. The actual link is established during processing, so that no other preparation is required.

XDMS file input and updating are accomplished via an easy to use facility which offers 'editor-like' control, plus field copy, repeat function, command strings and user defined macros, all invokes by a single keystroke. Tabular and vertical screen modes accomodate most systems including those with 'narrow' screen widths. The extended forms feature permits optional user defined control sequences (in assembler) which may be used to control inverse video or highlighting, underlining and other 'fancy' functions on the terminal and printer.

XDMS has been designed to provide the user with maximum functionality, and yet be easy to use. No programming experience is required since all processing functions are controlled by high-level non-procedural commands. Whether your needs are for simple list-oriented applications or complex multi-file database networks, XDMS is the answer!

XDMS Data Management System (5" or 8" disk)	$179.95
XPAK set of DMS utilities (requires XDMS)	$ 79.95
XDMS+ Data Management System plus DMS utilities	$249.95
XACC General Accounting system (requires XDMS)	$249.95
Also available!	
DMS2/VM Data Management System (5" or 8" disk)	$ 99.95
UTIL: Set of 20 General Utilities (Non-DMS)	$ 59.95

WESTCHESTER Applied Business Systems
Post Office Box 187, Briarcliff Manor, N.Y. 10510

All software is written in macro/assembler and runs under 6809 FLEX O/S. Terms: Check, Money Order, Visa or Mastercharge. Shipment first class. Add P&H $2.50 ($7.50 Foreign). N.Y. Res add sales tax. Specify 5 or 8". Select South East Media 800-338-6800 Consultation! 914-941-3552 (eves).

FLEX is a trademark of Technical Systems Consultants, Inc.

OMEGASOFT INDUSTRIAL STRENGTH PASCAL

OMEGASOFT PASCAL

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming more critical that a high level language be used. Find out why over 500 companies have switched to OmegaSoft Pascal for their demanding applications.

WHY PASCAL?

Pascal was designed to teach students how to write structured programs that are easy to read and maintain. In the past decade it has also proved to provide the same advantages in industrial applications.

EXTENSIONS

OmegaSoft has taken the Pascal framework and expanded the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The byte data type allows you to directly address bytes in memory or I/O devices. The common arithmetic operations can be used for bytes along with shift left, shift right, "and", "or", "eor", and complement operators. These operators are also available for integer and hex (2 byte unsigned) numbers.

LongIntegers are four byte signed numbers useful for extended range arithmetic commonly needed for machine control. Functions have been added to allow conversion between the various data types. Dynamic length strings allow complex text manipulation and allow effective interactive I/O.

Variables can be placed either on the data stack (default), at an absolute address in memory (for I/O), in base page, relative to the program counter (for constant tables), or defined in another module.

FEATURES

The compiler generates assembly language for assembly and linking to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a much smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

Full source code is included for the runtime library, the debugger, and other support utilities.

ISO COMPATIBILITY

OmegaSoft Pascal has been tested using the pascal Validation Suite. The Suite is a collection of over 400 Pascal programs designed to test the quality of Pascal Compilers and their runtime systems for compliance with the ISO (International Standards Organization) Pascal standard. OmegaSoft is the only supplier of 6809 native Pascal compilers that publishes this report in its instruction manual.

DEBUGGER

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. The debugger allows very fast turnaround for programs to be run on the host system.

TARGET SYSTEMS

The target system may be any 6809 system. No specific I/O devices are required. The output code is re-entrant and rom-able, perfect for single-board systems up to large development systems. There are no charges for use of the output of the compiler or the object of the runtime library in your products. 68000 target coming in 3rd Qtr. 1983.

HOST SYSTEM

The host system must be 6809 based and have at least 48K of ram (56K recommended) and run one of the following operations systems: MDOS, XDOS, OS-9, or FLEX. Priced from $425. 68000 host coming in 4th Qtr. 1983.

SUPPORT PRODUCTS

The OmegaSoft Relocatable Assembler and Linking Loader is designed to support the Pascal Compiler Package and can also be used for general assembly language program development. Priced from $125.

OmegaSoft's Screen Editor supports smart terminals and comes complete with the Pascal source. Priced from $90.

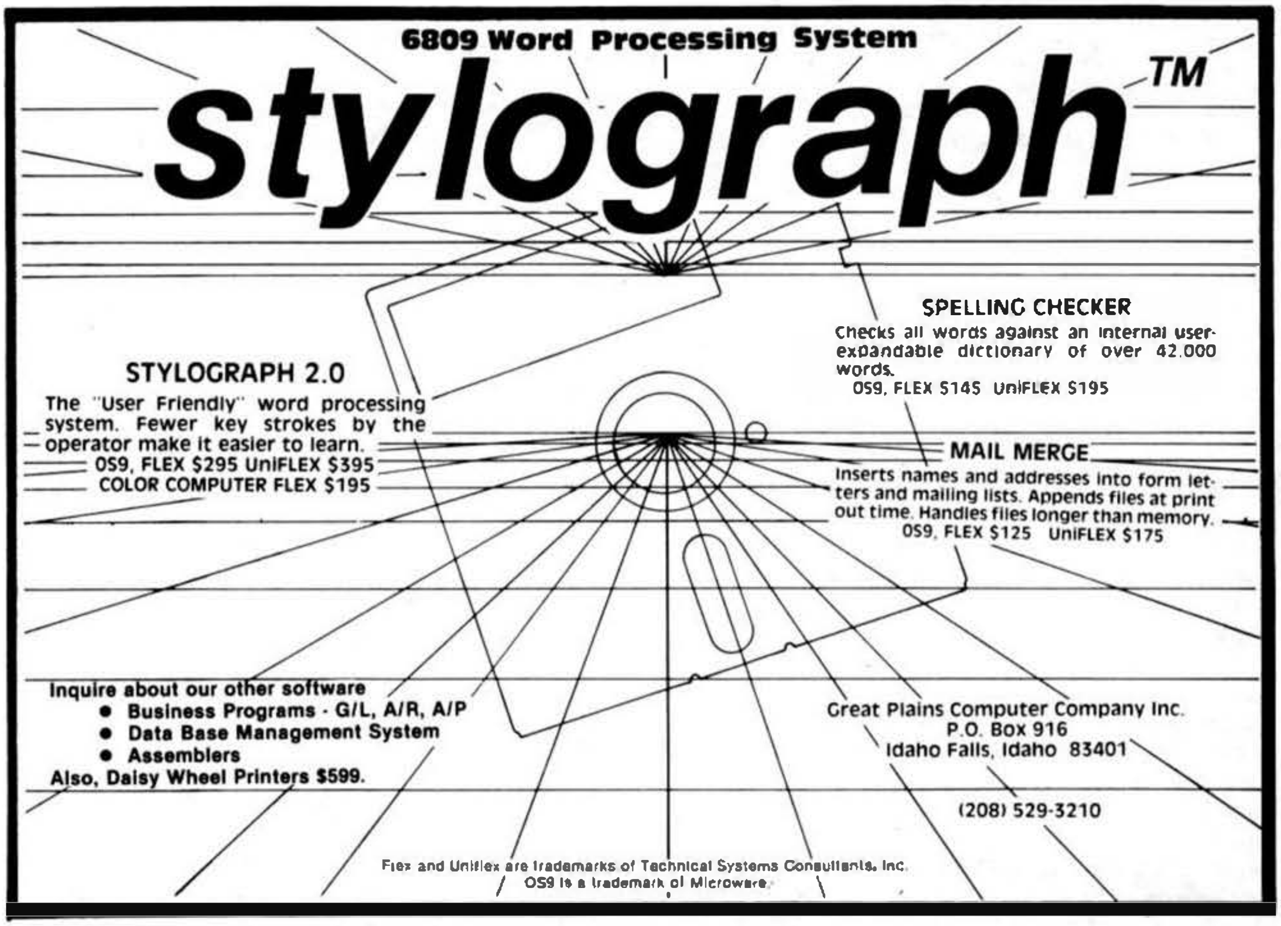
Faster floating point execution can be obtained by use of the Arithmetic Processor Unit option which uses the AMD9511 APU chip. Priced from $90.

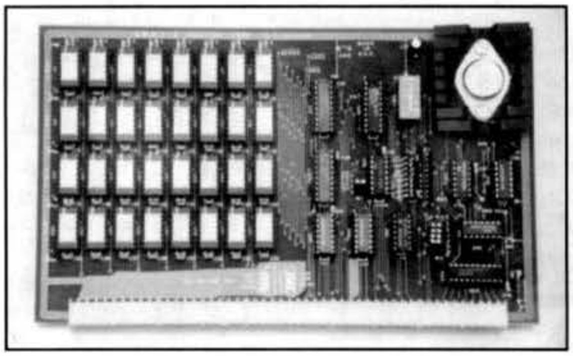
For complex real-time applications, the Multi-tasking Kernel provides task scheduling, inter-task communications, and resource interlocking. The Kernel is a runtime library that is accessible as Pascal functions and procedures (with full source included). Priced from $175.

Dealer and OEM inquiries invited. OmegaSoft products are also available from distributors in Australia and Western Europe, call or write for more information.

OMEGASOFT
P.O. Box 842
Camarillo, CA 93010
(805) 987-6426

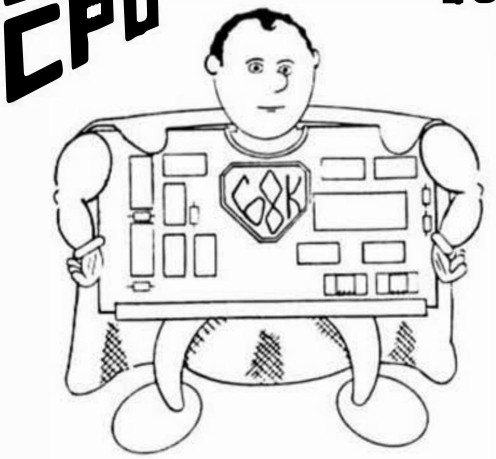
TM: MDOS and XDOS are Motorola trademarks. FLEX is a trademark of TSC. OS-9 is a trademark of Microware.





SUPER CPU

...only from LSI



Yes, it's SUPER-CPU, advanced, new 32 bit processor with power and ability far beyond those of normal CPU's. SUPER-CPU, who can change the course of mighty SS-50 processing, break 68XX benchmarks with its bare MC68008 CPU. And who, disguised as LSI's S68K/08-CPU SS-50c compatible card, fights a never ending battle for speed, power and the SS-50c user.

We're mighty proud of our new processor card. We're giving you the ability to go 68000 without major changes to your system. Our new CPU gives you these advanced features:

- Dynamic partitioning memory management unit with bound check register.
- On-board timer for multi-user/multi-tasking applications.
- On-board boot-strap EPROM and Monitor EPROM space.
- Vectored priority interrupt generator.
- On-board wait state generator.
- User selectable bus options that includes a new higher bandwidth bus mode.
- And many more...

S68K/08-CPU.............	$799.95
CP/M-68K..................	$350.00

(contact factory for disk controllers supported)

CP/M-68K is a registered trademark of Digital Research, Inc

VISA MasterCard

N.Y. residents add sales tax

LSI Enterprises Ltd.
PO Box 1227
Woodhaven, NY 11421
(212) 423-5596

68 MICRO JOURNAL PROGRAMS on DISK

Disk #1: FILESORT, MINICAT, MINICOPY, MINIFMS, **LIFETIME, **POETRY, **FOODLIST, **DIET.
Disk #2: DISKEDIT w/ Inst. & fixes, PRIME, *PRMOD, **SNOOPY, **FOOTBALL, **HEXPAWN, **LIFETIME.
Disk #3: CBUG09, SEC1, SEC2, FIND, TABLE2, INTEXT, DISK-EXP, *DISKSAVE.
Disk #4: MAILING PROGRAM, *FINDDAT, *CHANGE, *TESTDISK.
Disk #5: *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk #6: **PURCHASE ORDER, INDEX (Disk file Indx).
Disk #7: Linking Loader & RLOAD, Harkness
Disk #8: CRTSET, Lanpher (May '82)
Disk #9: DATECOPY, DISKFIX9 (Aug '82)

NOTE: All are as published or received by 68 Micro Journal, some have fixes and patches.

This is a reader service only! No Warranty is offered or implied, they are as received and are for reader convenience ONLY. Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk $19.95 - 5" Disk $17.95

68 MICRO JOURNAL
POB 794
Hixson, TN 37343
615-842-4600

* indicates 6800, ** indicates BASIC SWTPC or TSC - 6809 no indicator.

MASTER CARD - VISA accepted - Foreign add sufficient postage surface or air!!

ANDERSON COMPUTER CONSULTANTS & Associates

Ron Anderson, respected author and columnist for 68 MICRO JOURNAL announces the **Anderson Computer Consultants & Associates**, a consulting firm dealing primarily in 68XX(X) software design. Our wide experience in designing 6809 based control systems for machine tools is now available on a consultation basis.

Our experience includes programming machine control functions, signal analysis, multi-axis servo control (CNC) and general software design and development. We have extensive experience in instrumentation and analysis of specialized software. We support all popular languages pertaining to the 6809 and other 68XX(X) processors.

If you are a manufacturer of a control or measuring package that you believe could benefit from efficient software, write or call Ron Anderson. The fact that any calculation you can do with pencil and paper, can be done much better with a microcomputer. We will be happy to review your problem and offer a modern, state-of-the-art microcomputer solution. We can do the entire job or work with your software or hardware engineers.

Anderson Computer Consultants & Associates
3540 Sturbridge Court
Ann Arbor, MI 48105

'68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

THOMAS INSTRUMENTATION

HARDWARE	ASSEM & TEST	BARE CARD
CPU-5	$299.00	$49.00
S-R/R w/o memory chips	$120.00	$49.00
S-R/R with 48K	$399.00	$49.00
SP-1	$195.00	$49.00
6802 SUPER CPU	$235.00	$59.00
VIDEO RAM	$195.00	$49.00
PARALLEL I/O	$139.00	$49.00
SS-50/50C EXTENDER	$ 35.00	NA
SS-30 EXTENDER	$ 25.00	NA
SS-50 WIRE WRAP	NA	$39.00
SS-30 WIRE WRAP	NA	$20.00
SS-30 BACKPLANE 8POS.	NA	$39.00
SS-50 BACKPLANE 4,8,12, & 16 @ $5.00/SLOT	NA	NA

COMPONENTS

GOLD MALE MOLEX	$1.60	GOLD FEMALE MOLEX	$1.60
TIN MALE MOLEX	$ .40	TIN FEMALE MOLEX	$ .50
146805E2P	$20.00		

MASTERCARD, VISA, AND C.O.D. ACCEPTED. PLEASE ALLOW 3 WEEKS FOR CHECKS TO CLEAR. CONT. USA ADD $3.00 SHIPPING, CANADA $6.00, FOREIGN $12.00. NJ RESIDENTS ADD 6% SALES TAX
FLEX and UniFLEX ARE TRADEMARKS OF TECHNICAL SYSTEMS CONSULTANTS

SOFTWARE

VDISK
Use extended memory
as a fast disk drive
6809 source & obj $149.00
6809 object $ 99.00

OUTSIDE MODEM PROGRAM
includes source
UniFLEX™ $100.00
FLEX™ 6800 or 6809 $ 50.00

CROSS ASSEMBLER
For 6800, 6801, 6805
Runs on 6809 FLEX™
$150.00

THOMAS INSTRUMENTATION
168 EIGHTH STREET
AVALON, N.J. 08202
(609) 967-4280

CSG

Clearbrook Software Group Inc.

OS-9 SOFTWARE

Disk Utilities Package $49 (CSG-DUP) — Four utilities designed to save you time (and typing). DDir— display a directory listing of one or more directories or of an entire disk. **DCopy** — copy files in one or more directories or an entire disk with or without prompting. DDel — Delete files in one or more directories or an entire disk. **DAttr** — re-attribute files or change owner I.D. for files in one or more directories or an entire disk. All utilites allow wild card file specification (partial filename, attribute and own I.D.) and have the ability to use AND, OR and NOT logic for file specification relations.

DEdit $39 (CSG-DEDIT) — A screen oriented disk editor. DEdit allows the user to view and modify any readable sector on an OS-9 format disk. A terminal with clear screen and cursor addressing functions is required. Data is displayed in hexadecimal, ASCII and binary formats. Modifications to a sector are made by moving the cursor to the byte(s) to be changed, typing the correct data and issuing an Update command. (Written in machine language.)

BT9 $59 (CSG-BT9) — A binary tree filing program. BT9 makes it easy to write BASIC09 programs in which files must be maintained in sorted order. Three types of files are supported: Index File, Indexed Data File and Non-indexed Data File. BT9 also allows for maintaining several indexes for one file. All sorting is performed on disk leaving more memory free for other programs. BT9 is written in machine language for fast operation (typical search time under two seconds for a file of 2000 records).

SPECIAL OFFER — Order 2 CSG products and deduct 10%. Order all 3 products and deduct 15%. Any orders received before 83/09/30 will receive CrashCopy. CrashCopy attempts to copy files from a disk on which the disk allocation map sector is damaged.

Call (604) 853-9118 for information.

Specify 8 inch or 5 inch format.

Send Certified Cheque or Money Order to: Clearbrook Software Group, Box 8000, Suite 499, Sumas, WA 98295. In Canada write: Box 8000, Suite 499, Abbotsford, B.C. V2S 1K3

FIRST CLASS

INTELLICOM

INTELLigent COMunications Program

INTELLICOM provides you with the capability of **Intelligent** computer to computer **communications** from both a terminal emulation and file transfer standpoint. **INTELLICOM supports** several file **transfer** protocols that facilitate the transfer of both binary and ASCII data. Since **INTELLICOM** is **menu drive**, it is a breeze to use and understand. With **INTELLICOM** you will be able to **communicate** with the various data and **timesharing services** such as The Source and CompuServe. Additionally, since **INTELLICOM supports** the protocol used by virtually all remote **CP/M** systems around the world, all users can immediately begin to take advantage of the wealth of public domain sotware available on these systems. Current, or potential, users of CompuServe can transmit and **receive** both **binary** and **ASCII data** with full error detection and recovery. The checksum protocols allow for the verifications of data blocks transferred (assuming appropriate support on the host end). This feature will be of great value in those applications where data integrity is paramount. **INTELLICOM's documentation** includes a **detailed** description of all protocols used along with machine readable examples of host pseudo code. A telephone directory and dialing facility compatible with Hayes Smartmodems is supported.

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

Price: OS9 or FLEX Version $99.95, add $5.00 COD
JOTO ASSOCIATES, 104 Lepage Dr., Southington, CT 06489
203-621-8070

TO REALIZE THE FULL POWER & PERFORMANCE OF THE 6809, LOOK TO GIMIX.

GIMIX OFFERS YOU A VARIETY OF SS50 BUS COMPONENTS AND SYSTEMS.

OS-9 GMX III

The GMX 6809 CPU III and OS-9 GMX III. A Multi-user, Multi-tasking package for the ultimate in System Performance plus protection of the system and other users from crashes caused by errors in individual users programs

#01 (CPU & Software) NEW!	$1698.01

INTELLIGENT I/O PROCESSOR BOARDS increase system throughput by reducing interrupts to the host, buffering data transfers, and data preprocessing. Prices include on board firmware. Requires system drivers.

#11 3 port RS232 Serial (SS30) NEW!	$498.11
#12 4 port Parallel (SS50)	$638.12
OS-9 GMXIII drivers .. (included when purchased with GMX III package)	$200.00

OS-9 Level 2 users – contact GIMIX for system requirements and availability.

192K GMX III #79 SYSTEMS: All include GMX 6809 CPU III and OS-9 GMX III (#01); a #11 3 port Intelligent serial I/O & cables; #19 Classy Chassis; 192KB Static RAM, #68 DMA controller, all necessary cables, power regulators, and filler plates. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are included.

#79 with dual 40 track DSDD drives NEW!	$5998.79
#79 with dual 80 track DSDD drives	$6298.79
#79 with #88 8" Dual Drive Disk System	$7698.79
#79 with #90 19MB Winchester subsystem & one 80 track DSDD drive	$8998.79

UniFLEX for the GMX 6809 CPU III and intelligent I/O boards is in development.

OS-9 GMX I; OS-9 GMX II; FLEX; and UniFLEX

The **#05** GIMIX 6809 PLUS CPU board			$578.05
Options: GMX DAT	$35.00	SWTPC DAT	$15.00
9511A	$312.00	9512	$265.00

#49 64KB GHOST SYSTEM includes: #05 CPU; #19 Classy Chassis; 64KB static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX I. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are also included.

#49 with dual 40 track DSDD drives	$4398.49
#49 with dual 80 track DSDD drives	$4698.49
#49 with #88 8" Dual Drive Disk System	$5998.49
#49 with #90 19MB Winchester subsystems & one 80 track DSDD drive	$7398.49

#39 128KB SY TEM includes: #05 CPUwDAT; #19 Classy Chassis; 128KB of static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX II. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB, and GMX-VDISK for FLEX are included.

#39 with dual 40 track DSDD drives	$4998.39
#39 with dual 80 track DSDD drives	$5298.39
#39 with #88 8" Dual Drive Disk System	$6598.39
#39 with #90 19MB Winchester subsystem & one 80 track DSDD drive	$7998.39

UniFLEX, available at extra cost, requires 8" or Winchester drives. A signed license agreement with TSC is required before shipment.

You can add to any GIMIX system RAM, I/Os and other options, or substitute non-volatile RAM. GIMIX will customize to your needs.

COMING SOON: Contact GIMIX for price and availability on 40MB and 72MB Winchester (5¼") drives, removeable pack Winchesters, 256KB static RAM boards.

All GIMIX systems are guaranteed for 2MHz operation. GIMIX systems include documentation for all boards and software in a GIMIX binder. **ALL DRIVES ARE 100% TESTED AND ALIGNED BY GIMIX.**

ALL BOARDS AND SYSTEMS ARE ASSEMBLED, BURNED IN, AND TESTED. GOLD-PLATED BUS CONNECTORS ARE USED.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033. Visa or Master Charge also accepted.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX Inc. reserves the right to change pricing, terms, and product specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Choose from GIMIX' wide variety of system components.

The **GIMIX CLASSY CHASSIS #19** consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board			$1398.19
Triple Disk regulator card and cables	$88.22	Baud rate generator card	$88.93
Missing cycle detector	$38.23	Filler plates	$14.92
Back panel connector plates (specify)	$8.60	50 Hz. option	$30.00

MEMORIES (GIMIX uses only Static RAM)

#67 64KB NMOS STATIC RAM board	$478.67
#64 64KB CMOS STATIC RAM board w/battery back-up	$568.64
#34 8K PROM board	$98.34
#32 16 socket PROM/ROM/RAM board	$238.32

I/O Boards (see above for Intelligent I/Os)

#41 Single port serial, RS232/20ma. current loop	$88.41
#43 2 port serial, RS232	$128.43
#46 8 port serial, RS232	$318.46
#42 2 port parallel	$88.42
#45 8 port parallel	$198.45
#50 serial, RS232, RS422, RS423	$244.50
#52 SSDA serial, RS232, RS422, RS423	$254.52
#54 ADLC serial, RS232, RS422, RS423	$268.54
Each cable with connectors for back panel mounting (specify board)	$24.95

DISK CONTROLLERS

#68 DMA (featured in all systems above)	$588.68
#28 dbl. dens. programmed I/O (5" drives only)	$298.28
#58 single dens. programmed I/O (5" and/or 8" drives)	$226.58
#48 same as #58 but for 5" drives only	$198.48
Cable sets: 8" with Back Panel connector	$29.25
for two 8" external drives	$44.28
for two 5" drives	$34.98

SOFTWARE: GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #68 controller.

When ordered with any controller, FLEX is			$30.00
GMXBUG PROMs and manual			$98.66
Boot or Video boot PROM	$30.00	UNIFLEX boot PROM	$50.00
OS-9 GMX I	$200.00	OS-9 GMX II	$500.00
Editor	$125.00	Assembler	$125.00
BASIC-09	$200.00	RUNB	$100.00

DISK DRIVES FOR GIMIX SYSTEMS – complete with cables and power regulators.

5" DSDD 40 track			2 for $900.00
5" DSDD 80 track			2 for $1300.00
#88" Dual 8" DSDD drives, cabinet, power supply, & cables			$2698.88
Cabinet only	$848.18	220V 50Hz. Option, add	$30.00
Filler plate	$14.83	Cable for 2 drives	$44.82
Cable for 4 drives	$67.84	Cable for cabinet to mainframe	$45.81

WINCHESTER SUBSYSTEMS: for use only in GIMIX systems with #68 DMA controller.

#90: includes one 19MB drive, Interface, and Software	$3588.90
#91: includes two 19MB drives, Interface and Software	$5288.91

Contact GIMIX for price and availability of other forthcoming subsystems.

OTHER BOARDS

#76 GHOST 80X24 VIDEO BOARD			$398.76
#66 50 pin Protoboards	$56.66	**#33** 30 pin Protoboards	$38.33
#03 6800 CPU			$224.03
#06 6800 CPU with timers	$288.06	Baud rate option, add	$30.00
#08 RELAY DRIVER (board, bracket, transformer, and 31 relays)			$1128.08
#86 - #08 (board, bracket, transformer, without relays)			$538.86
#85 OPTO board			$348.85
WINDRUSH EPROM PROGRAMMER			$375.00
3" Binder	12.00	2" Binder	$9.00

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

DON'T SEE IT??? ASK! OUR BROCHURE HAS MORE COMPLETE DESCRIPTIONS AND SPECS. PHONE OR WRITE TODAY FOR YOUR COPY.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX inc.
1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

DATA COMP has everything you need to make your TRS-80C Color Computer WORK for YOU: from Parts and Pieces to Full Ready To Use Systems. DATA-COMP designs sells services and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea)	$150.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller	
when purchased with Special General FLEX9 Sys.	$ 49.95
when purchased without the General FLEX9 Sys	$ 59.95
Color Computer with 64K RAM and EXT BASIC	$399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER Radio Shack COLOR DISK CONTROLLER, a Disk Drive System Special General Version of FLEX9, F-MATE(RS), and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set

$999 95

64K Radio Shack COLOR COMPUTER Radio Shack COLOR DISK CONTROLLER, a Disk Drive System and a Box of 10 Double Density Diskettes, a COMPLETE ready to run SYSTEM on your Color TV Set

$799 95

DISK DRIVE PACKAGES, etc.

These Packages include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable:

PAK #1 - 1 Single Sided, Double Density Sys.	$489.95
PAK #2 - 2 Single Sided Double Density Sys.	$749.95
PAK #3 - 1 Double Sided, Double Density Sys.	$559.95
PAK #4 - 2 Double Sided, Double Density Sys.	$919.95
PAK #5 - 2 Qume Thinline Double Sided, Double Density Sys.	$699.95
PAK #6 - 2 Tandon Thinline Single Sided, Double Density Sys.	$599.95

PRINTERS

EPSON RX-80	$375.00
EPSON MX-100	$725.00
EPSON MX-80	$395.00
EPSON FX-80	$599.00

SERIAL BOARDS

MX Series	$119.95
RX FX-Series	$ 99.95
Spectrum MX-Series	$ 54.95
Mark Data Keyboards	$ 67.95

FLOPPY DISKETTES

	Scotch	Verbatim
5" Soft Sector Disks		
Single Sided Single Density	$2.40ea.	$2.75ea
Single Sided Double Density	$2 40ea	$2.75ea
Double Sided Double Density		$4.92ea
Plastic Storage Box		$2.00ea
8" Soft Sector Disks		
Single Sided Single Density		$3.75ea
Single Sided Double Density		$4.10ea
Double Sided Double Density		$4.75ea
Plastic Library Box		$5.00ea

PARTS AND PIECES

Radio Shack Disk Controller	$169.95
1 Single Sided Double Density Disk Drive Tandon	$229.95
1 Double Sided Double Density Disk Drive Qume	$319.95
1 Qume Thinline Double Sided, Double Density	$249.95
Single Drive Cabinet with Power Supply	$ 79.95
Double Drive Cabinet with Power Supply	$109.95
Single Drive Disk Cable for RS Controller	$ 19.95
Double Drive Disk Cable for RS Controller	$ 24.95
Micro Tech Prods Inc. LOWER CASE ROM Adapter	$ 74.95
Radio Shack BASIC Version 1.1 ROM or 1.2 ROM	$ 34.95
Radio Shack Extended Basic ROM	$ 74.95
Radio Shack Disk Basic ROM 1.1	$ 29.95
Screen Clean - Clears Up Video Distortion On Your Color Computer	$ 39.95
Set of Eight 64K RAM Chips w Mod Instructions	$ 49.95

For Ordering Call TOLL FREE 1-800-338-6800

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W)
Intelligent Terminals

New & Demo Models

At Discount Prices

Remaining Supply of
SWTPC 8212 CRT Terminals

EM-82 Video Terminal

Emulates the 8200 Series from SWTPC

DATA-COMP

P.O. Box 794 HIXSON, TN 37343
1-615-842-4601

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

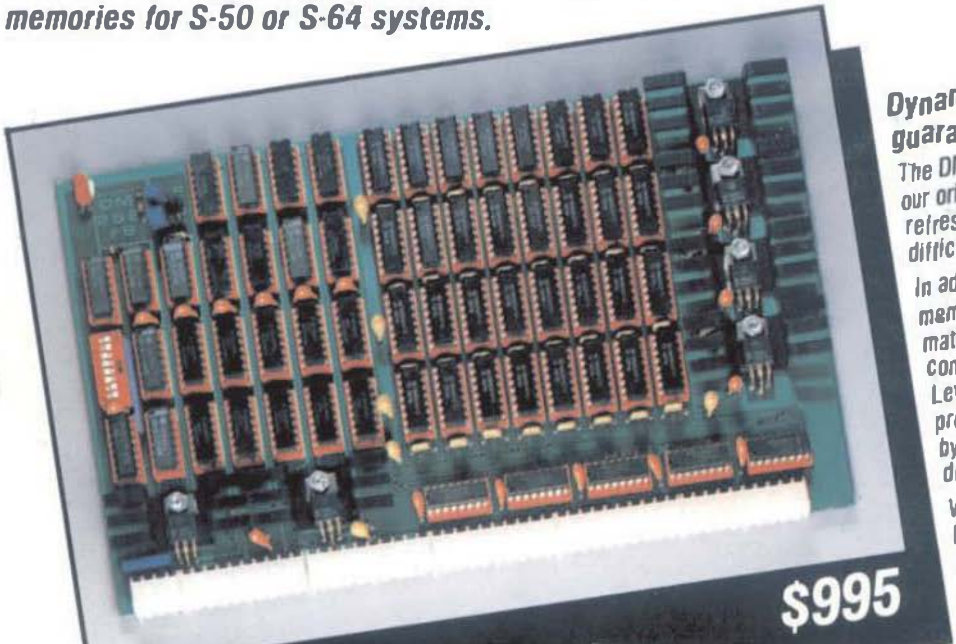
Data-Comp—South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

MEMORY • MEMORY • MEMORY • MEM

Hazelwood Computer Systems presents a pair of advanced memories for S-50 or S-64 systems.



$995

256K

Dynamic memory with guaranteed 2.5 MHz operation

The DM-256 utilizes the same proven refresh method as our original DM-64. This technique allows fully transparent refresh while maintaining full bus speed and eliminating difficult timing problems.

In addition, individual power regulators for each row of memory ICs enhance reliability by providing the ultimate in noise decoupling. Flexible addressing mode controls make the entire 256K available to OS9 Level 2 or UniFLEX. For FLEX, the furnished utility program HYPERDSK,™ formats memory not used by FLEX as a virtual, superfast disk, thus tremendously improving total system performance.

With several hundred already in use, the DM-256 is the best choice for economical, reliable memory.

Assembled, tested, and burned in

Order: DM-256

Static memory with on-board battery backup

Utilizing the latest in low power CMOS static RAMs, the CM-64 has an on-board NiCad battery to maintain memory contents in the absence of bus power. Generous timing margins allow operation at bus speeds in excess of 2 MHz. Zener diode power loss sensing inhibits write operations to prevent data loss during power transitions. Write access can be explicitly controlled by means of an on-board jumper, a remote mounted switch, or an external logic signal.

This allows the memory to appear as ROM to the computer after the contents have been "programmed" using normal computer operations. The economical price makes the advantages of non-volatile memory available to any system.

Assembled, tested, and burned in

Order: CM-64

64K

$395

HAZELWOOD COMPUTER SYSTEMS

907 East Terra, O'Fallon, MO 63366, 314-281-1055

HELIX™

HYPERDSK is a trademark of Hazelwood Computer Systems • OS9 is a trademark of Microware • FLEX and UniFLEX are trademarks of TSC